CHARACTERIZING ACOUSTIC ENVIRONMENTS WITH OLAF AND ELSA

A Thesis
Presented to
The Academic Faculty

by

Brandon T. Carroll

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Electrical and Computer Engineering

Georgia Institute of Technology
August 2018

CHARACTERIZING ACOUSTIC ENVIRONMENTS WITH OLAF AND ELSA

Approved by:

Dr. David V. Anderson, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Aaron D. Lanterman
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Mark A. Davenport
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Duen H. Chau
School of Computational Science and Engineering
Georgia Institute of Technology

Dr. Marilyn C. Wolf
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Wayne Daley
Georgia Tech Research Institute
Georgia Institute of Technology

Date Approved: July 19, 2018

ACKNOWLEDGEMENTS

Although a Ph.D. is often viewed as personal achievement, in reality it results from the collective effort of a large number of people, most of whom work quietly behind the scenes. While I cannot list them all here, I could not have reached this milestone without their support.

First and foremost, I am grateful to my parents Quinn and Margaret Carroll for the countless ways they have taught, inspired, and supported me throughout my life.

Dr. David Anderson has been an excellent advisor and mentor throughout my years at Georgia Tech. The bulk of the ideas presented here stemmed from discussions with him.

This research was done in collaboration with Georgia Tech Research Institute. Dr. Wayne Daley has been closely involved in directing the project, and has been like a second advisor to me. Dr. Doug Britton and Sim Harbert also dedicated many hours to the project. Lucy Johnson provided invaluable administrative support.

I have had the pleasure of working and interacting with many other current and former members of the ESP Lab. In particular, I would like to recognize Dr. Bradley Whitaker, Dr. Muhammad Rizwan, Nathan Parrish, Lee Richert, and Dr. Kaitlin Fair. They all influenced this work in various ways, and also provided friendship and support along the way.

Finally, I would like to thank the State of Georgia, the Georgia Poultry Federation along with its member companies, and the Agricultural Technology Research Program at Georgia Tech for supporting this work. I would also like to thank the Poultry Diagnostic Research Center and the Departments of Poultry Science at the University of Georgia and at the University of Arkansas for their support in collecting data. Drs. Mark Jackwood, Jeanna Wilson, Maricarmen Garcia, and Karen Christensen were especially helpful.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

The confluence of signal processing and machine learning has created many innovative technologies in popular research areas such as speech recognition. However, many of the most successful methods are difficult to apply in areas that are not as popular and lack institutional support for research and creation of labeled data corpora. For instance, the expertise, computational resources, and quantities of data required for deep learning create a significant barrier to its implementation in many fields that it might otherwise impact. The focus of this work is the development of signal processing and machine learning methods that can be practically implemented with less human effort, less need for large quantities of labeled data, and less computational cost. Toward this goal, we have developed methods for outlier learning using augmented frozen dictionaries (OLAF) and estimating the likelihood of sparse approximations (ELSA) in the context of monitoring acoustic environments. Both methods utilize sparse, dictionary-based representations to capture information about the structure of the data. These methods are potentially applicable in many different areas and to many different types of data, but in this work have been tested specifically for monitoring poultry production facilities. The high levels of noise and uncontrolled nature of these environments pose significant challenges for audio signal processing efforts, especially when combined with a lack of labeled data. Our methods have proven effective at leveraging randomly selected unlabeled data and weakly labeled data to characterize the environment and highlight events or changes in the conditions in poultry houses. Providing these types of tools for monitoring livestock could help producers to better understand the effects different practices have on the animals and lead to better animal well-being.

CHAPTER 1

INTRODUCTION

1.1 Motivation

In recent years, signal processing and machine learning algorithms have helped advance several areas of research and have led to the release of innovative new products. Speech recognition, automatic photo tagging, and music recommendation engines are among the most successful. However, there are many other unexplored domains that could potentially derive great benefit from these algorithms and techniques.

This work develops signal processing and machine learning techniques for acoustic environmental monitoring within the poultry industry. Billions of chickens are raised in the U.S. each year [81], and application of these types of algorithms could lead to a number of benefits. These include detecting and responding to problems more rapidly, developing a better understanding of how different conditions affect animal well-being, and giving producers better ways to respond to increasing consumer demands regarding animal well-being. With up to 25,000 birds in each growout house, even small improvements can have a significant financial impact.

Farmers periodically send a few of their birds to a lab to be tested for disease and health issues. However, they do not currently have cost-effective ways of continuously monitoring the birds at the farm. Instead, workers are sometimes advised to "sit quietly, watching and listening to the birds" in order to detect signs of sickness [105]. Allowing the birds to acclimate to a human's presence and settle down makes it easier to hear low gurgling and other quiet noises that are signs of common avian diseases such as infectious bronchitis. Workers may or may not be able to recognize such symptoms depending on their experience and knowledge. Other diseases (e.g. laryngotracheitis) may have more readily noticeable symptoms, such as coughing and sneezing.

In addition to health, the concept of well-being extends to questions of how different

environments or conditions affect the happiness of the birds. Because there are no known ways to directly measure the happiness of a bird, opinions on best practices vary greatly in this area. However, there are indications that characterizing the vocalizations of farm animals could provide insight into the internal state of the animals [73].

Unlike speech recognition, the use of signal processing algorithms on poultry audio data is not a well-established domain. Thus, there are a number of challenges to doing research on poultry audio. For example, speech recognition researchers benefit from large corpora of labeled data that have been collected over many years and can be used to train and test their algorithms. Such datasets are not available for poultry, and almost all of our data comes from our own collection efforts. Speech recognition research also benefits from language models and our general understanding of the language being spoken. We do not have language models for poultry, and our understanding of the meaning of the different types of noises chickens make is limited. Because of this, labeling individual sounds in poultry audio data is difficult and somewhat subjective.

Additional challenges arise due to the nature of poultry farming. Significant amounts of background noise are introduced by the equipment in the facility (e.g. ventilation fans and feed augurs) and by the chickens themselves (e.g. movement and general chatter). The characteristics of this noise vary as equipment turns on and off and as the behavior of the chickens changes at different times of day.

Some of the types of conditions we would like to be able to detect, such as disease, are extremely rare because farmers work hard to prevent them. Although we do have some data on diseases collected in controlled research settings, data on diseases from actual commercial settings is currently unavailable. Furthermore, the data we do have is imbalanced because the audible symptoms of disease, such as coughs, are only manifested sporadically.

1.2 Contributions

This research develops signal processing and machine learning techniques that can be applied practically to monitor environmental audio, with a focus on the application of poultry monitoring. Our primary concern in determining whether or not a system is "practical"

is the amount of human effort required to get it to produce useful results in new settings or applications. Many established techniques can produce very good results given copious amounts of labeled training data. However, the amount of human effort required to produce sufficient labeled data is often prohibitive, especially in smaller domains that do not have large corporations driving the development of such systems.

The challenges we face in making such algorithms practical are open questions in signal processing that are common across many domains:

- How to make use of large amounts of unlabeled data?
- How to best leverage very small amounts of labeled data?
- How to deal with heavily imbalanced data?
- How to deal with significant background noise?
- How to extract important information without models or foreknowledge of what is important?
- How to adapt algorithms to new environments with minimal human effort?
- How to deal with natural changes or drift in the data (day/night cycles, feeding, aging, etc.)?
- How to make use of weak labels that specify that a certain condition is present in a broad time range, but not exactly when it is present?

We believe that the methods of outlier learning via augmented frozen dictionaries (OLAF) and estimating the likelihood of sparse approximations (ELSA) developed herein can help address many of these questions.

This research also contributes to the field of bioacoustics. Although some prior work has been published on detecting different types of stress in chickens, we were unable to find other bioacoustics work on disease detection with chickens or on generally characterizing the acoustic environment in chicken houses over long periods of time. Also, much of the other work in bioacoustics does not operate on continuous recordings, but instead relies on

manual extraction (and sometimes filtering) of the sounds that are passed into the recognition algorithm. Of the publications that deal with continuous recordings, many only look at hours of audio at a time. We consider continuous recordings spanning weeks and months at a time, where variables such as day/night cycles and the aging of the animals become more relevant.

1.3 Organization

This document is organized as follows: Chapter 2 describes the various datasets we collected for our experiments and reviews the related bioacoustics literature. Chapter 3 details how we calculate features from the audio waveforms and reviews other related methods. Chapter 4 gives an overview of earlier disease detection methods that we explored using standard machine learning algorithms. Chapter 5 details the outlier learning via augmented frozen dictionaries (OLAF) method and its improved results for disease detection scenarios. Chapter 6 develops our method of estimating the likelihood of sparse approximations (ELSA) and evaluates its ability to highlight abnormal events in the data. Chapter 7 summarizes the contributions of this work and lists ways it might be expanded in future work.

CHAPTER 2

DATA

2.1 Data collection

Several sets of data have been collected to support this research. All of our datasets involving disease come from the Poultry Diagnostic and Research Center (PDRC) at the University of Georgia at Athens (UGA). Researchers at PDRC develop and test vaccines in controlled experiments, providing an excellent opportunity for us to record both healthy and sick chickens.

We have also collected data from healthy chickens from other experiments at the UGA research farm, from a single growout cycle in a commercial broiler house, and from commercial flocks raised at the University of Arkansas's research farm. Several of the datasets we have collected are summarized in Table 1 and detailed in the following sections.

2.1.1 Arkansas data (Ark1)

The research farm at the University of Arkansas owns four commercial-style growout houses in which commercial broiler flocks are raised. We instrumented one of their houses with ten Raspberry Pi based recording systems spaced throughout the house. The Ark1 dataset represents the first of the flocks we recorded in that setting.

2.1.2 Breeder data

The breeder data was recorded from a flock of breeders housed in a long pen at the UGA research farm. Two microphones, weatherboards, and cameras were placed at either end of the pen and recorded the chickens over a period of 233 days, making it our longest continuous dataset.

2.1.3 Commercial data (COM)

A commercial grower allowed us to install our system in a growout house and record a flock over the entire six week growout period. The house contained roughly 25,000 broilers,

Table 1: Datasets

Dataset abbr.	Number of birds	Duration (days)	Notes
Ark1	∼25,000	42	The first of six different commercial flocks recorded in the research farm at the University of Arkansas
Breeder	Several hundred	233	Breeder chickens recorded during an experiment at UGA
COM	∼25,000	51	Recorded in a commercial growout house
IB	6 per box	25	Experiment group infected with infectious bronchitis (IB)
LT1	35–78	15	All birds infected with laryngotracheitis (LT)
LT2	89-126 per room	20	Three out of four rooms infected with LT
LT3	64–103 per room	15	Three out of four rooms infected with LT
Temp1	∼200	61	Six heat stress tests performed

and was equipped with two microphones and two weatherboards hung about 20 ft apart. Everything operated normally and the chickens were healthy throughout the six weeks.

2.1.4 Infectious bronchitis data (IB)

We have collected data from one IB vaccine trial performed at PDRC. The IB data consists of continuous recordings made from two isolation boxes (see Figure 1) over a period of 25 days. Six day-old broilers were placed in each of the boxes, with one box serving as the control group and the other as the experiment group. The experiment group was challenged (infected) with the IB virus on day 15, and clinical signs were observed on day 22 confirming that the chickens had gotten sick. Each box was equipped with a microphone, a camera, and a weatherboard with temperature and humidity sensors (see Figure 2).

The Merck Veterinary Manual describes IB as an "acute, highly contagious disease of major economic importance in commercial chicken flocks throughout the world." IB has nearly 100% morbidity (almost every chicken exposed to the virus becomes sick) and typically about a 5% mortality rate without other complications. The disease can cause pink eye, impaired breathing, lower feed consumption, and reduced weight gain. Egg production

Figure 1: The two isolation boxes that housed the control and experiment groups during the recording of the IB data.

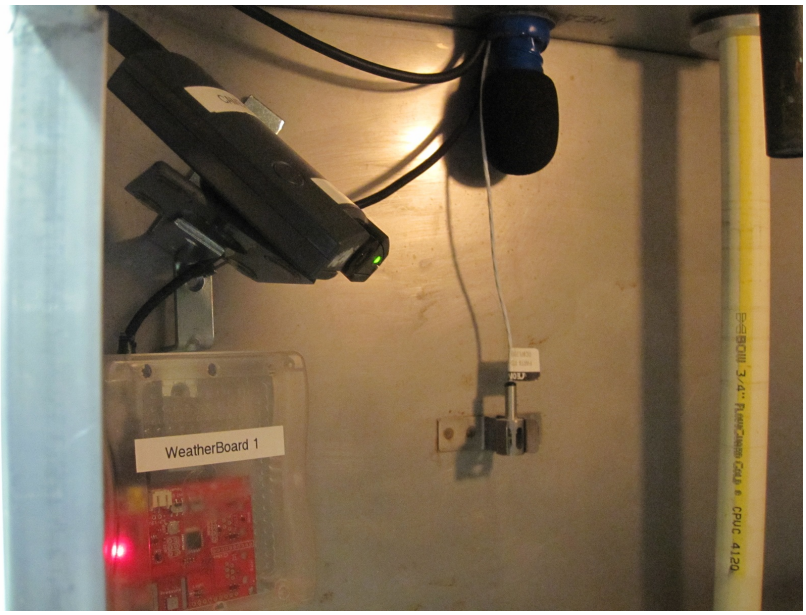

Figure 2: The sensors installed inside one of the isolator units during collection of the IB data. The microphone is mounted through the ceiling, and the camera and weatherboard are on the left.

in layers can drop as low as a third of the normal rate, and the eggs that are laid are often low quality and misshapen. It can take up to 8 weeks for egg production levels to return to normal, and chickens may shed the virus for as long as 20 weeks after infection [5].

The audible respiratory signs caused by IB are often what tip farmers off that there is a problem with a flock. In particular, IB is characterized by a sound called a rale, which is a gurgling noise produced as the chickens try to breathe through an excess of mucus produced in the trachea. Rales can be very quiet and subtle, making them difficult to notice above background noises—especially for people unfamiliar with what they sound like. Although rales are also symptomatic of other respiratory diseases like laryngotracheitis, they are particularly prevalent in the case of IB. Because of this, our IB related work has been focused primarily on the detection of rales. Since rales do not occur in healthy chickens, their presence is an almost certain indication that there is a problem.

2.1.5 Laryngotracheitis data (LT)

We recorded three different experiments at PDRC involving LT. The LT1 data contains recordings from a single room starting with 35 broilers. The birds were 35 days old when the experiment began and were infected the following day. Strong clinical signs were observed on day 6, and 33 additional birds were introduced to the room on day 7 to test if the disease would be transmitted. The original 35 birds were removed on day 10. Two microphones were hung in opposite corners of the room and recorded continuously throughout the experiment.

The LT2 and LT3 data both contain recordings from 4 rooms. One of the rooms received a lower dosage of the vaccine under trial, while another room received a higher dosage of the vaccine. The remaining two rooms served as the negative control (not infected) and positive control (infected, but not vaccinated). Each room was equipped with a microphone, a camera, and a weatherboard. PDRC staff recorded clinical signs during the 4 or 5 days when the birds were expected to be the most sick.

According to *The Merck Veterinary Manual*, LT is an acute, highly contagious infection that has been reported in most areas in the U.S. with intensive poultry farming. LT causes severe breathing problems, coughing, rales, and loss of appetite. Affected birds become

inactive, and as many of half of them may pass away from the disease. Symptoms typically show up 5 to 12 days after exposure and last for about two weeks. In laying flocks, it also causes decreases in productivity. Birds remain carriers of the disease and can infect other birds for life, even after recovery. The disease is controlled by implementing biosecurity measures and administering vaccines [5].

2.1.6 Temperature test data (Temp1)

A flock of chickens were raised in a room at UGA's research farm over a period of 61 days. Towards the end of the data, six heat stress tests were performed on each of six consecutive days. These tests involved raising the temperature in the room by about 15–20 degrees Fahrenheit for about two hours.

2.1.7 Other datasets

Several other datasets were collected in the course of this work, but are not included in the results presented in this thesis. All of them have been analyzed using the methods presented in Chapter 6. Their results are generally consistent with those presented in this work and are omitted for brevity.

2.2 Labeling

Many machine learning algorithms require labeled data for training purposes. There are two approaches we have taken to labeling data for training. The first is to manually label individual sounds in the recordings, such as rales and coughs. The second is to label broader periods of time as, for instance, healthy or sick.

2.2.1 Labeling individual sounds

Labeling individual sounds is labor intensive and requires someone with experience or familiarity with the types sounds being labeled. We consulted with poultry science experts at PDRC for guidance on identifying relevant sounds so that we could then label data for training. Even with the aid of experts, noises present in the recordings can sometimes be ambiguous because the various types of sounds made by chickens are not fully discrete. Sometimes the sounds are combinations of things, such as a cough and a chirp. Ambiguity

is also introduced when sounds from multiple birds or other sources overlap. Despite these challenges, labeling the data in this way has the benefit of allowing us to train classifiers to recognize specific sounds that we know are symptomatic of a disease. This gives us more confidence that the algorithm is actually detecting symptoms of the disease instead of some other potential confounding factor.

To label individual sounds, we use the open source program Audacity. This allows us to view a spectrogram while listening to the audio and labeling specific segments. The labels can then be exported to a tab-separated text file containing the starting time, the ending time, and the text of each label. These files can then be parsed and used by our processing code. Since our label starting and ending times do not line up perfectly with the time windows over which we calculate features, we assign a label to a feature vector only if the the labeled time period overlaps with more than 50% of the processing window.

We have not made extensive efforts to label large amounts of data because one of the goals of this work is to develop techniques that are easy to apply to new situations with minimal human effort. However, we did label small amounts of data containing rales (from the IB dataset) and coughs (from the LT datasets) as part of our earlier efforts detailed in Chapter 4.

2.2.2 Situational labeling

Labeling broad periods of time when a condition (such as sickness) was present is much easier and quicker than picking out individual vocalizations. Our recording system includes a logging interface that we ask the chickens' caretakers to use, which gives us some general information on when certain conditions are present. A drawback to labeling broad periods of time is that it is more difficult to guarantee that the algorithms are not learning to recognize some other confounding factor, such as the age of the chickens or a specific type of background noise. This difficulty can be mitigated if healthy and sick data collected under a variety of circumstances are available. Unfortunately, the need to carefully control and contain diseases limits the circumstances under which we can collect sick data.

2.2.3 Leveraging unlabeled data

The datasets listed in Table 1 represent thousands of hours of recording, and the cost of manually labeling all of them would be prohibitive. However, unlabeled data can still be useful for algorithms that attempt to learn the patterns or structure within the data. Chapters 5 and 6 include methods that attempt to do this while still being able to take advantage of any labels that might be available.

2.3 Prior bioacoustics work

Bioacoustics is the study of the production, perception, and interpretation of sounds produced by living organisms. Previous work in this area has motivated and guided our work with chickens. For instance, researchers have noted an urgent need for non-invasive methods to measure farm animal welfare. They also indicated that vocalizations could be used to make inferences about the well-being of animals if interpreted correctly [73].

2.3.1 Chickens

Most of the bioacoustics work that has been done with chickens focuses on different types of stress. Canterbury et al. measured call characteristics of Leghorn laying hens under stress from hunger, thirst, and heat. Differences in several of the characteristics were observed between the different types of stress and normal conditions [17].

Otu-Nyarko performed a series of experiments exploring the effects of different types of stress on chickens' vocalizations. The stressors included heat, crowding, human presence, human shouting, handling, and a few combinations of the aforementioned. Vocalizations were manually extracted from the recordings and filtered to remove background noise. A short time Fourier transform was then taken to get spectral features over each time window. The spectral features were clustered using a Gaussian Markov model (GMM), and the resulting sequence was passed to a hidden Markov model (HMM) for classification. Overall accuracies for the different experiments typically lied in the 70%–80% range. Otu-Nyarko also reported evidence that age, breed, and even diet can cause differences in vocalizations [86, 85].

Colón evaluated how well a number of different audio features correlated with stress from environmental conditions, including temperature, humidity, and ammonia levels. The data was bandpass filtered prior to feature calculation to help get rid of background noise from fans and other equipment. Of the features tested, kurtosis and loudness were the best for detecting stress. An AdaBoost classifier with a decision stump as the weak classifier attained accuracy percentages in the low 90s using these two features [36].

Parrish developed an algorithm that estimated the age of broiler chicks based on their vocalizations. Cepstral coefficients were passed into a C4.5 decision tree for classification, yielding age estimates that were typically within a day of the actual age of the chicks [87].

Curtin found the number of broiler vocalizations to be an effective indicator of heat stress. He used spectral oversubtraction to remove noise, then set a threshold within the expected frequency range to detect vocalizations. The vocalizations were then counted over a sliding window and thresholded to yield perfect detection of stress conditions. Curtin notes that this detection method is specialized for the circumstances under which the data was collected, and likely would not generalize well to other situations [39].

Recently, Lee et al. used a series of one-against-all SVMs to classify layer hen vocalizations as being under normal conditions, cold stress, heat stress, or mental stress (induced by hitting the cage with a stick). After manually extracting and labeling sound samples, they calculated 54 different audio features and used correlation based feature selection to choose the most useful eight. They reported an accuracy of 96.2% [63].

We are unaware of prior bioacoustics work that deals specifically with the detection of diseases in poultry. Furthermore, much of the work that has been done for detecting stressful conditions has relied on sound samples manually extracted by the researchers. This effectively cuts out most of the potential distractors that would be encountered for continuous monitoring in a commercial setting. These distractors include noise from things such as flapping wings, worker activity, drinker strikes, feeding, equipment turning on and off, etc. Such algorithms are not feasible for implementation in commercial settings if they require human workers to extract the specific sounds to feed in to the system. Our work contributes to the state of the art both by addressing disease detection and by enabling

more practical continuous monitoring.

2.3.2 Other organisms

Several other species have been studied by bioacoustics researchers. Work has been done with pigs to detect screams [75, 107], stress [74, 79], coughs [25], and pig wasting disease [32]. Research efforts with cows have included efforts to translate calls into English meanings [52] and to develop systems that detect cows' estrus periods using MFCCs [31] and formant structure [64]. Matos et al. detected human coughs from continuous recordings by extracting events above an energy threshold and passing them into HMMs [76]. Many studies have sought automatic classification of bird species and bird song using MFCCs [28, 46, 56] and various other (often hand-picked) features [27, 48, 101, 106]. Lee et al. applied MFCCs and linear discriminant analysis to classify the calls of many different species of frogs and crickets [60]. Kohlsdorf developed a spectral pattern recognition engine for dolphin vocalizations [58].

The Automated Remote Biodiversity Monitoring Network (ARBIMON) created by Aide et al. has established a network of several permanent audio recording stations in wilderness areas. These stations upload recordings to the project website, which hosts an interface for browsing and training HMM recognizers. The project has yielded long-term data on the vocal activity of the fauna in different areas, including birds, insects, frogs, and mammals [4].

As part of the Doolittle project, Clemins et al. developed Greenwood function cepstral coefficients (GFCCs) and generalized perceptual linear prediction (GPLP), which are generalizations of MFCCs and PLP that allow them to be adapted for species that vocalize and hear in different frequency ranges. Work from this project included classification of vocalizations from beluga whales [33], African elephants [33, 34, 35], ortolan buntings [35, 95], Asian elephants [95], and chickens [95]. Brown and Smaragdis later used GMMs and HMMs to classify killer whale vocalizations [14].

CHAPTER 3

FEATURES

3.1 Introduction

The raw waveforms of audio recordings are rarely passed directly into learning algorithms because they contain far more information than is relevant for most tasks. Instead, a set of features are typically derived from the waveform and then are fed into subsequent algorithms. Feature calculation attempts to summarize the relevant information while discarding irrelevant information, and is the first step in all our algorithms.

3.2 Feature types

Many different types of features have been developed over the years for audio data. This section summarizes a small subset of those feature types, focusing on the ones that are either used in this work or are closely related to it.

3.2.1 Mel-frequency cepstral coefficients (MFCCs)

MFCCs are a set of features that can be computed from audio data and are designed to match how humans perceive sound. MFCCs are based on the mel scale, which relates the frequency of sound to a measure of the perceived pitch. The mel scale was originally developed by Stevens et al. in 1937 [103], and later refined in 1940 [102]. A few different formulas have been fit to the data that defines the mel scale. One of the most commonly used ones was given by O'Shaughnessy as

$$m = 2595 \log \left(1 + \frac{f}{700} \right), \tag{1}$$

where f is the frequency in hertz and m is the corresponding frequency in mels [84]. This is the formula we use when calculating MFCCs in this work.

The word "cepstral" in MFCCs is obtained by reversing the first four letters of the word "spectral," and refers to the cepstrum originally defined by Bogert [9]. A cepstrum is

obtained by applying a frequency-based transform (e.g. the Fourier or cosine transform) to the log power spectrum of a signal. For example, a cepstrum could be computed as

$$\textit{cepstrum} = \mathrm{DCT}\Big\{\log\Big(|\mathcal{F}\{x(t)\}|^2\Big)\Big\}, \tag{2}$$

where $\mathrm{DCT}\{\cdot\}$ denotes the discrete cosine transform, $\mathcal{F}\{\cdot\}$ denotes the Fourier transform, and $x(t)$ is the signal. The second application of a frequency-based transform has a tendency to decorrelate the resulting features, which can be helpful for many machine learning algorithms. A mel-scaled cepstrum inserts a triangular filter bank whose frequency bands are linearly spaced in the mel domain immediately prior to the log operation.

Mermelstein was the first to apply mel-scaled cepstra to a speech recognition problem [77], although he attributes the idea to Bridle and Brown [13]. MFCCs were later shown to be superior to several other feature types used for speech recognition because of their ability to represent perceptually relevant aspects of the sound [40]. After this, MFCCs grew to become one of the most commonly used features in speech recognition. Jankowski et al. later demonstrated that other, more complex front ends for speech systems provided little gain over MFCCs even though they required significantly more computation [53]. Sandhu and Ghitza showed that the inclusion of dynamic features (deltas and delta-deltas) for MFCCs provided significant performance gains under all the different signal conditions they tested [97].

The process we use to calculate MFCCs can be summarized as follows:

1. Apply a pre-emphasis filter of the form $x_i = x_i - ax_{i-1}$ to the waveform to compensate for spectral tilt. We used the value $a = 0.97$.

2. Calculate the short-time Fourier transform (STFT) of the audio waveform. Discard phase information and keep the power. We used a window size of 25 ms and a shift of 10 ms.

3. Window and warp the frequency axis using triangular windows linearly spaced along the mel scale.

4. Take the logarithm of the results.

5. Calculate the discrete cosine transform (DCT) to help decorrelate the resulting features.

6. Optionally calculate deltas and delta-deltas.

The DCT step near the end turns the resulting MFCC features into linear combinations of energies from disparate frequency bands. This makes it difficult to intuitively understand what the individual MFCC values represent, other than the 0th MFCC which is related to the total energy present in the signal.

3.2.2 Greenwood function cepstral coefficients (GFCCs)

GFCCs, developed by Clemins et al. [35], are a generalization of MFCCs that take into account frequency-cochlear measurements for the species under consideration. When these measurements are not available, parameters can be approximated using the hearing range of the species. Chickens have a similar hearing range to humans, so we have deemed MFCCs appropriate in this case.

3.2.3 Log-mel energies

Log-mel energies are computed in the same way as MFCCs (see Section 3.2.1), except that they omit the DCT step at the end. This makes them more intuitively understandable because they directly correspond to the amount of energy in different frequency bands.

Log-mel energies are one of the most commonly used feature types in the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge. In 2017, the top-ranked algorithms for all the different DCASE tasks used log-mel energy features.

3.2.4 Mel magnitude spectrum

Mel magnitude spectrum features are inspired by log-mel energies, but with modifications intended to better preserve additivity. This is a desirable property because there is an assumption of additivity inherent in the sparse coding algorithms that we use. The modifications are to use the magnitude spectrum instead of the power spectrum, and to omit the log scaling at the end of the log-mel energy computation.

The magnitude spectrum is technically not additive because waveforms added together may cancel each other out to some degree, resulting in spectral magnitudes smaller than the sum of the spectral magnitudes of the component waveforms. However, in practice and with real, non-adversarial audio sources, the magnitude spectrum tends to behave closely enough to additive for our purposes.

The algorithm we present in Chapter 6 incorporates a log-scaling after sparse coding has been performed. This gives it a stronger similarity to the commonly used log-mel energies, but with the sparse coding step inserted before the operations that more egregiously violate the assumptions of additivity.

3.2.5 Delta features

Estimates of the first and second derivatives of a set of features—typically called deltas and delta-deltas—are often included with the original features, especially when MFCCs are used. Since the deltas and delta-deltas capture information about how the underlying features are changing over time, they are also often referred to as dynamic features. Including these dynamic features has been shown to improve various recognition tasks [97].

We calculate the deltas as

$$\Delta[t] = \frac{\sum_{n=1}^{N} n(y[t+n] - y[t-n])}{2\sum_{n=1}^{N} n^2}, \tag{3}$$

where $y[t]$ represents the value of a given feature for the tth sample and N is the number of samples on either side of the current sample to use in estimating the derivative. This formula represents the slope of a least-squares linear regression over the computation window. For the first and last N samples of a signal, the computation window will extend past the beginning or the end of the signal. In these cases, we repeat the corresponding first or last sample in place of all the missing samples. Delta-deltas are computed by applying Equation (3) a second time. Also note that Equation (3) is a linear operator, and thus does not violate the assumptions of additivity inherent in the sparse coding algorithm.

Most of our work with the ELSA method in Chapter 6 uses features that include only the deltas and delta-deltas of the mel magnitude spectrum. The derivative operation acts like a high-pass filter, and we have found it to be effective in filtering out much of the

background fan noise present in the audio from poultry houses. Considering only the delta and delta-delta features helps our algorithms respond more to the activity of the chickens and less to the background noise.

In this work, we used $N = 4$ to estimate the deltas and delta-deltas over a nine-sample window. The original features, deltas, and delta-deltas of our poultry data tend to fall in different magnitude ranges, causing one to dominate the others when used together in dictionary learning and sparse coding. To prevent this, we scaled the delta-deltas by a factor of 5 to bring them on par with the deltas. When the original mel magnitude spectrum features were also included, we scaled both the deltas and delta-deltas by an additional factor of 8.

3.3 Dimensionality reduction

Although feature calculations can be viewed as a form of dimensionality reduction, they are typically not tuned to a specific situation unless the features are designed by hand. Passing features through a dimensionality reduction algorithm can help extract the information most relevant to the current task, which can speed up learning and reduce overfitting.

3.3.1 K-means

Some of our preliminary work makes use of the k-means algorithm. K-means is a simple, popular clustering algorithm that transforms multidimensional data into cluster indices. The standard algorithm alternates between assigning observations to the nearest cluster centroid and updating the cluster centroids based on the new membership. K-means was developed by Lloyd [68], although its name was coined by MacQueen [69]. The algorithm can be sped up by using k-d trees for the nearest neighbor search [55], and results can be improved with intelligent initialization [12].

3.3.2 Sparse representations

Most of this work uses sparse representations for dimensionality reduction. Olshausen and Field discovered that passing natural images into an unsupervised learning algorithm with a sparsity-based objective function produced receptive fields whose characteristics match

those of the mammalian primary visual cortex [82]. Expanding upon this work, they later observed that when an overcomplete dictionary was used, the sparse coding algorithm would only recruit the most pertinent dictionary elements for a given data sample—a non-linear behavior similar to that of cortical simple cells [83]. The discovery of this link between sparsity and biological perception helped spark interest in sparse representations.

Sparse representations seek to decompose a signal into a linear combination of underlying components. The problem is regularized by an assumption that only a few of the components should be active at a time. The sparse coding problem can be formulated as

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_0 \quad \text{subject to} \quad \boldsymbol{y} = \mathbf{D}\boldsymbol{x}, \tag{4}$$

where $\boldsymbol{y}$ is the target signal, $\mathbf{D}$ is the dictionary matrix, and $\boldsymbol{x}$ is the coefficient vector. This problem has been proven to be NP-hard [80]. However, it has also been shown that if perfect reconstruction of $\boldsymbol{y}$ is possible with a sufficiently sparse $\boldsymbol{x}$, then the solution of

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{y} = \mathbf{D}\boldsymbol{x} \tag{5}$$

is unique and equal to the solution of Equation (4) for almost all $\mathbf{D}$ [41]. Changing the ℓ^0-norm to an ℓ^1-norm turns the problem into a convex optimization problem, whose solution can be found with various optimization tools. Further theoretical work has shown that under certain conditions, greedy coding algorithms can recover the exact solution to the sparse coding problem [104] and that they can still work well in the presence of noise [42]. Although the conditions under which these theoretical results hold often cannot be guaranteed for real data, sparse representations have empirically proven useful in several applications [16, 24, 51, 109].

We leverage sparse representations in our approach to novelty detection and classification problems. When applied to acoustic features, the sparse coefficient matrix can act like a map of what types of sounds are present at what time. Ideally, each dictionary element would map to a different source of sound. In real life, the representation is not nearly so clean. Variations in the sounds produced and imperfections in the learning and coding algorithms tend to yield a many-to-many relationship between sound sources and the coefficients used to

represent them. However, the sparse coding still provides insight into the structure present in the data.

3.3.2.1 Sparse coding algorithms

Given a dictionary, pursuit algorithms can be used to find a sparse code for a signal. Matching pursuit iteratively projects the remaining error (or residue) onto the dictionary elements and greedily subtracts off the one that matches best until some stopping criterion is met [72]. However, this fully greedy approach can lead to over- or under-subtraction of components in earlier stages since the components are not orthogonal. The orthogonal matching pursuit (OMP) algorithm addresses this problem by ensuring that the residue is orthogonal to all the selected components after each iteration [88]. Various modifications of OMP have been proposed to improve its speed and performance [43, 96, 100]. Other sparse coding algorithms include the focal underdetermined system solver [47] and variants of basis pursuit [15, 26, 50].

3.3.2.2 Alternating minimization

Charles modified Olshausen's unsupervised dictionary learning approach to be able to enforce non-negativity of the coefficients and applied it to hyperspectral imagery [24], using Koh's interior point method to solve the ℓ^1-regularized optimization problem [57]. The sparse coding model was able to learn the spectral signatures of the various materials in the scene and was able to improve the performance of a subsequent supervised classification algorithm for hyperspectral imagery—particularly when the labeled training sets were very small.

We used Charles' dictionary learning algorithm [24] for part of our work in Chapter 5. We refer to his learning algorithm as "alternating minimization" because it alternates between optimizing the dictionary and the associated coefficient matrix as it seeks to minimize the reconstruction error. In this respect, his algorithm operates similarly to expectation-maximization (EM) algorithms. The alternating minimization algorithm is summarized in Algorithm 1.

Algorithm 1 The alternating minimization algorithm [24], where each $\boldsymbol{y}_m$ is a data vector (a column of $\mathbf{Y}$), each $\boldsymbol{x}_m$ is a sparse coefficient vector (a column of $\mathbf{X}$), $\mathbf{D}$ is the sparse coding dictionary, λ is a fidelity-sparsity tradeoff parameter, $\mathcal{C}$ is the set of matrices in $\mathbb{R}^{\mathrm{N}\times\mathrm{K}}$ whose columns have ℓ^2-norm less than one, and M is the number of data samples selected for training. Algorithm adapted from [18].

Input: Signals $\{\boldsymbol{y}_m \in \mathbb{R}^{\mathrm{N}}\}_{m=1,...,\mathrm{M}}$, initial dictionary $\mathbf{D}_0 \in \mathcal{C}$, regularization term λ, number of iterations K

1: Initialize $\mathbf{D} \leftarrow \mathbf{D}_0$
2: **for** $k = 1, ..., \mathrm{K}$ **do**
3: **for** $m = 1, ..., \mathrm{M}$ (*in parallel*) **do**
4: Calculate coefficient vectors:
5: $\boldsymbol{x}_m = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{y}_m - \mathbf{D}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$
6: **end for**
7: Update dictionary:
8: $\mathbf{D} = \arg\min_{\mathbf{D}\in\mathcal{C}} \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \frac{1}{2} \|\boldsymbol{y}_m - \mathbf{D}\boldsymbol{x}_m\|_2^2$
9: **end for**
10: **return D**

3.3.2.3 K-SVD

K-SVD is a partially greedy dictionary learning algorithm that can serve as a more computationally efficient alternative to Charles's method. K-SVD, developed by Aharon et al. [2], operates by updating each of the dictionary elements and its corresponding coefficients while holding the rest of the dictionary and coefficients constant. These updates seek to minimize the objective function

$$\min_{\mathbf{D},\mathbf{X}} \left\{ \|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_{\mathrm{F}}^2 \right\} \quad \text{subject to} \quad \forall i, \|\boldsymbol{x}_i\|_0 \leq t_0, \tag{6}$$

where $\mathbf{Y}$ is the data matrix, $\mathbf{D}$ is the dictionary matrix, $\mathbf{X}$ is the sparse coefficient matrix, $\boldsymbol{x}_i$ is the ith column of $\mathbf{X}$, and t_0 is the sparsity constraint. The K-SVD algorithm is given in Algorithm 2. Note that the dictionary and coefficient updates in lines 3–12 never change which coefficients are non-zero in the rows of $\mathbf{X}$. Thus, any changes in the support of a dictionary element must come through the sparse coding step in line 2. The use of singular value decomposition (SVD) to minimize Equation (6) is appropriate because the Frobenius norm is equal to the sum of the squares of the singular values.

Rubinstein implemented an optimized version of the K-SVD algorithm that replaces the

Algorithm 2 The K-SVD algorithm to learn dictionaries for sparse representations [2], where $\mathbf{D}$ is the dictionary matrix, $\mathbf{Y}$ is the data matrix, and $\mathbf{X}$ is the coefficient matrix. Matlab-like notation is used to index and slice matrices. Algorithm adapted from [18].

Input: $\mathbf{D}, \mathbf{Y}$ ▷ Initial $\mathbf{D}$ can be randomly chosen, normalized columns of $\mathbf{Y}$
Output: $\mathbf{D}, \mathbf{X}$

repeat
  $\mathbf{X} \leftarrow \textsc{SparseCoding}(\mathbf{D}, \mathbf{Y})$ ▷ OMP is the recommended algorithm
  for all $\boldsymbol{d}_k$ in $\mathbf{D}$ **do** ▷ Random ordering; $\boldsymbol{d}_k$ is the kth column of $\mathbf{D}$
    $\boldsymbol{\ell} \leftarrow \{i \mid \mathbf{X}(k, i) \neq 0\}$ ▷ Get the locations of non-zero values in row k of $\mathbf{X}$
    $\mathbf{Y}_\mathrm{R} \leftarrow \mathbf{Y}(:, \boldsymbol{\ell})$ ▷ Slice out columns of $\mathbf{Y}$ where $\mathbf{X}(k, :) \neq 0$
    $\mathbf{X}_\mathrm{R} \leftarrow \mathbf{X}(:, \boldsymbol{\ell})$ ▷ Slice out the same columns of $\mathbf{X}$
    $\mathbf{X}_\mathrm{R}(k, :) \leftarrow 0$ ▷ Zero out the kth row of $\mathbf{X}_\mathrm{R}$
    $\mathbf{E}_\mathrm{R} \leftarrow \mathbf{Y}_\mathrm{R} - \mathbf{D}\mathbf{X}_\mathrm{R}$ ▷ Error for the selected columns when $\boldsymbol{d}_k$ is not used
    $\mathbf{U}\mathbf{S}\mathbf{V}^\top \leftarrow \mathrm{SVD}(\mathbf{E}_\mathrm{R})$ ▷ Only the largest s.v. and its vectors are needed
    $\mathbf{D}(:, k) \leftarrow \mathbf{U}(:, 1)$ ▷ Update the kth dictionary element
    $\mathbf{X}(k, \boldsymbol{\ell}) \leftarrow \mathbf{S}(1, 1)\mathbf{V}(:, 1)^\top$ ▷ Update its non-zero coefficients
  end for
until stopping criterion

SVD computation with a fast approximation and uses his optimized Batch-OMP algorithm for the sparse coding step [96]. Throughout this work, we used his implementation in all places where we refer to K-SVD.

Aharon et al. also developed a non-negative variant of the K-SVD algorithm, NN-K-SVD [3], which uses the non-negative basis pursuit algorithm given by Hoyer [50] for the sparse coding step. Other options for dictionary learning include further variants of K-SVD [65, 96, 100], the method of optimal directions [44, 100], and others [59, 66, 70, 71].

3.3.3 Alternative methods for dimensionality reduction

Principal component analysis (PCA) reduces the dimensionality of data by projecting it onto a small set of orthogonal bases that maximizes the amount of variance preserved through the projection. The idea was originally presented by Pearson [89], and later expanded upon by Hotelling [49]. The PCA algorithm relies on the assumption that the directions of greatest variance contain the most useful information, which may not always be true.

Independent component analysis (ICA) seeks to decompose a signal into the sum of individual components, much like sparse representations. However, ICA uses the assumption

that the components will be statistically independent for regularization instead of an assumption of sparsity. ICA was originally addressed by Jutten and Herault [54] and applied to a blind source separation (BSS) problem. Choi et al. provide a review of the various different methods, applications, and extensions of BSS and ICA algorithms [29].

Non-negative matrix factorization (NMF) is another decomposition method developed by Lee and Seung [61, 62] that is related to ICA, BSS, and sparse representations. Of particular note, Smaragdis presented a method that is similar to the frozen dictionary approach we present in Chapter 5, but which he applied using NMF [98] and probabilistic latent component analysis [99] instead of sparse representations. He found it to provide excellent separation of vocals and a piano accompaniment [98]. NMF relies on the assumption that the inner dimension between the dictionary and coefficient matrices is small (relative to the other dimensions) in order to prevent the problem from being underdetermined. This effectively limits the number of dictionary atoms that may be used. For the poultry monitoring application, this limitation is problematic because we would like to have enough dictionary atoms to be able to represent a wide range of sounds from different sources.

While several of these other methods could potentially be applied in this work, sparse representations seem to provide the most natural fit for modeling the acoustic environment inside poultry production facilities. The OLAF and ELSA methods presented in Chapters 5 and 6 are both based on sparse coding. A comparison of the ELSA method against a modified method that uses PCA instead of sparse coding is given in Section 6.9.

CHAPTER 4

PRELIMINARY DISEASE DETECTION METHODS

Disease detection was one of the earlier thrusts of this work. This chapter covers a few preliminary experiments to evaluate the tenability of disease detection using standard machine learning techniques. The results here are later improved upon by the OLAF method in Chapter 5.

4.1 Rale detection

As described in Section 2.1.4, rales are a sound that chickens make as a symptom of common respiratory diseases such as infectious bronchitis (IB). Since rales do not occur with healthy chickens, detecting them can tip farmers off to outbreaks of disease in their flocks. This section describes an earlier algorithm for detecting rales in the IB dataset. Its contents are largely taken from [21], where the work was published.

4.1.1 Data labeling

With the guidance of experts at the Poultry Diagnostic and Research Center (PDRC), rales were manually labeled in various recordings from the experiment group in the IB dataset for use as training data. This task was difficult and time-consuming because many files contained rales that were barely discernible above the background noise. To avoid excessive false positives caused by including very quiet rales in the training data, we limited our training set to six files we had found with more clearly discernible rales and two files recorded prior to infection that contained no rales. All eight of these files came from the experiment group, so the training data did not include any data from the control group. After transforming the training data into the feature space, there were 629 samples marked as rales out of 15,976 samples total.

Our data was labeled using Audacity as described in Section 2.2.1. The results of our classification algorithm could be output as Audacity label files, with adjacent detections

merged into single labels. This provided a convenient way to visualize the performance of the approach on sample data.

4.1.2 Method

Using the training data, our detection algorithm was developed as follows:

1. Calculate MFCCs (including the deltas and delta-deltas) using a 25 ms window width and a 10 ms shift, yielding 39-dimensional vectors for each time slice. (See Sections 3.2.1 and 3.2.5.)

2. Cluster the vectors of MFCCs into 60 clusters using the k-means algorithm, yielding a single cluster index for each time slice. (See Section 3.3.1.)

3. Take histograms of the cluster indices over a 100 ms (or 8 sample) wide window and with a 30 ms (or 3 sample) shift.

4. Train a decision tree using the Weka's implementation of the C4.5 algorithm with a confidence factor of 0.002 and a minimum of 5 samples per leaf.

MFCCs were designed to match human auditory perception and are widely used in speech recognition [92]. Since humans are able to perceive rales, MFCCs provided a reasonable starting point for our choice of features as well as a significant reduction in dimensionality. We ran k-means on the MFCCs to automatically divide the samples into perceptually different clusters of sounds [69]. Because of the small window size of MFCCs, a single sample only contains information about what is happening during a short instant within a rale. Thus, we took a histogram of the cluster indices to gather information about the types of sounds that were happening over the duration of time that a short rale might last. Finally, we trained a decision tree to examine this histogram and determine if the distribution of sounds match those that would be expected in a rale. The decision tree parameters were manually tuned to yield a small tree to help ensure that overfitting was not occurring, given the small number of rales in our training data. Our final decision tree contained 10 leaf nodes.

Table 2: Training data confusion matrix for rale detection. Results taken from [21].

	Algorithm label	
Human label	None	Rale
None	15230	117
Rale	306	323

The C4.5 decision tree learning algorithm is Quinlan's extension of his iterative dichotomiser 3 (ID3) algorithm [94] to add support for missing values, continuous attributes, and attributes with different costs [93]. Like ID3, C4.5 attempts at each step to split the data in a way that maximizes information gain.

4.1.3 Evaluation

The confusion matrix obtained by running ten-fold cross validation on our training data is given in Table 2, and represents 73.4% precision and 51.4% recall. Although the training files were chosen because they contained relatively loud rales, there were still many quieter rales mixed in that were not detected. This contributed to the low recall value. Also, many of the misclassifications were caused by disagreement between the algorithmic labeling and the human labeling on when rales started and ended, even when they both detected the same rale. This mismatch in starting and ending times can be seen in Figure 3. For the detector to be useful in practice, it only needs to detect enough rales over time to make accurate decisions about the health of the chickens.

We tested our rale detection algorithm by running it on the full 25 days of recorded data for both the experiment group and the control group. On a 2.66 GHz Core II Quad CPU, it took only a few seconds to run the detection algorithm on each minute-long recording. The detection rates per day are shown in Figures 4 and 5. The detection rate of the experiment group clearly ramps up a few days after infection and remains high until the experiment ends, matching the course of the disease. Note that the chickens were removed at 8AM on the final day, but the recording continued through the remainder of the day. Thus, it is expected that the detection rate for day 25 would be about a third as high as the preceding

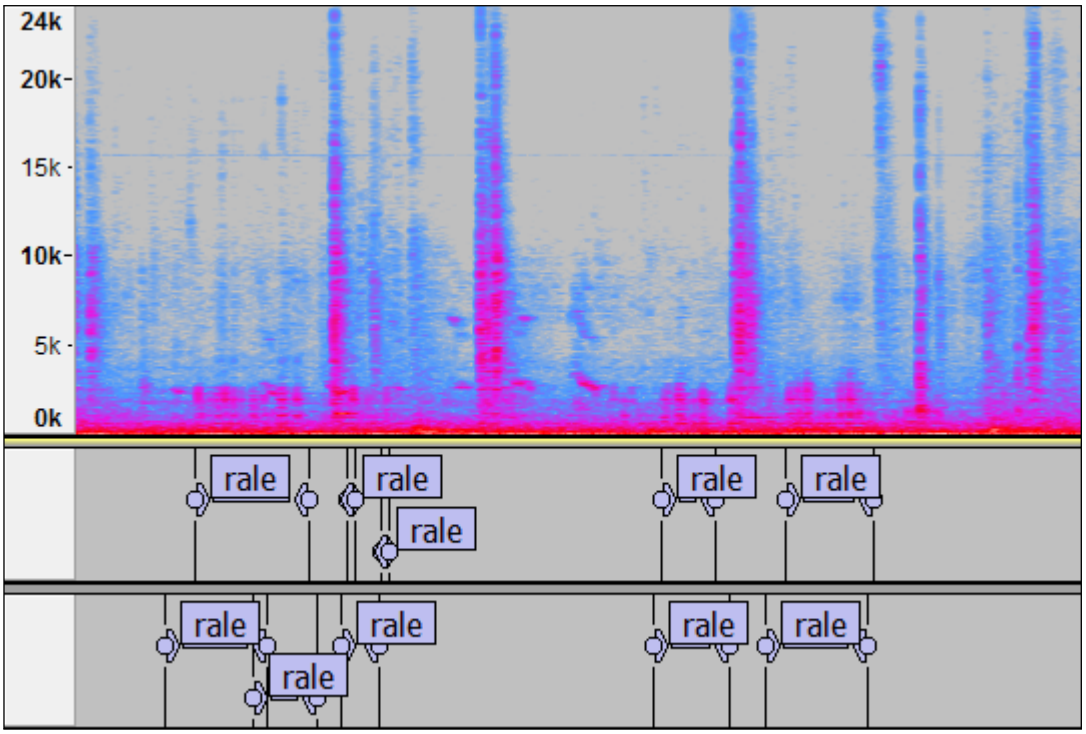

Figure 3: A comparison of human labeling and our algorithm's labeling of rales in one of the training samples, as viewed in Audacity. The top pane shows a spectrogram of the recording with frequency along the y-axis and time along the x-axis. The middle pane shows the human labeling of rales, and the bottom pane shows our algorithm's labeling. Although the same rales are often labeled by both, some of the misclassifications arise simply because the starting and ending times do not match up exactly. Figure taken from [21].

days, as reflected in the plot in Figure 4.

Although there are always some false positives, the detection rate remains low when the experiment group is healthy in Figure 4 and throughout all of the control group recordings in Figure 5. The control group does show a slight increase in detection rate over the same days as when the experiment group was sick, even though it was healthy during that period. This is likely because all of the training data that contained rales came from the sick period, which may have biased the detector toward age-dependent characteristics of the chicken sounds, or toward any background noises that were audible in both boxes during that time.

We located the 20 files with the highest detection rates for both the experiment and the control groups and had our classifier generate label files for them. This allowed us to import the labels into Audacity and examine the sounds that triggered the detections. All 20 files from the experiment group contained rales that were being detected correctly. The files from the control group contained various sounds that triggered false positives. Most were caused by a faint mechanical tapping noise present in several of the recordings that sounded very similar to short rales. Others were caused by workers talking and making noise in the room, or by occasional lower pitched vocalizations from the birds. These sounds occurred mostly in the same frequency range as rales, and were not represented in the small amount of training data.

4.1.4 Conclusion

Although this algorithm produces some misclassifications, it successfully detects enough rales to easily distinguish between times when the chickens are sick and when they are healthy. The results are encouraging in establishing the feasibility of automating detection of rales, especially because the background noise in the boxes used for our experiment is much louder than the typical noise levels in commercial poultry houses.

However, this algorithm cannot be easily transferred to new environments with different types of background noise. Running it on other datasets that do not contain rales produced large numbers of false positives, and it would be costly to label the data necessary to retrain it for each new environment. The small number of birds in the boxes also eliminates the

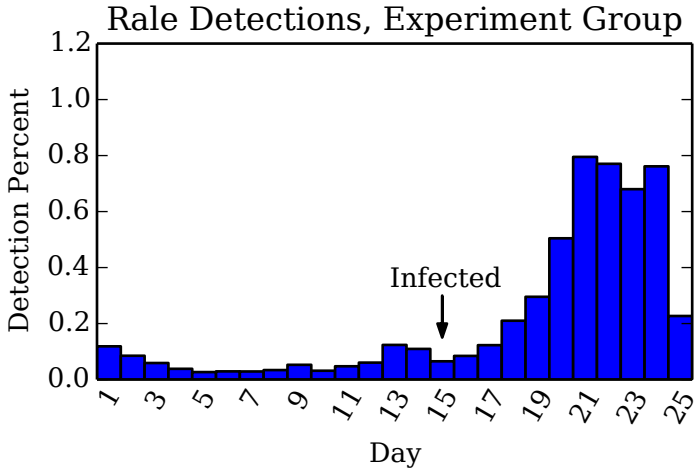

Figure 4: The rale detection rate per day for the experiment group. These chickens were infected on day 15 and were confirmed to be exhibiting clinical signs of the disease on day 22. The chickens were removed from the chamber about 1/3 of the way through day 25. The detection rates match the expected development of the disease. Figure taken from [21].

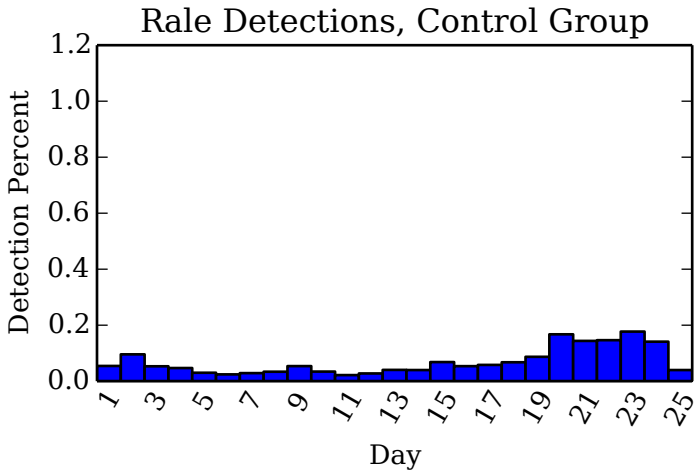

Figure 5: The rale detection rate per day for the control group. These chickens were healthy throughout the entire experiment, so no rales should be present at any time. The detection rate increases slightly over days 20–24, corresponding to when the experiment group was sick. This is likely a small bias in the algorithm caused by all of the training data that contained rales coming from that same time period and age range. Figure taken from [21].

constant chatter problem encountered in large houses with tens of thousands of birds.

4.2 Infectious bronchitis detection

Another method we explored sought to label each minute-long recording as either sick or healthy. This method was primarily developed by Bradley Whitaker and was published in [108]. The features used were derived by training a sparse coding dictionary (using the alternating minimization algorithm from Section 3.3.2.2) on a band of frequencies from the spectrogram, and then summing the resulting coefficients over the duration of a minute long recording. This process is shown in Figure 6. A support vector machine (SVM) [38, 11, 91] was then trained to label the summed coefficients as either healthy or sick, yielding the detection rates plotted in Figure 7.

As can be seen in the plots, this algorithm performed well in distinguishing between when the chickens were sick and when they were healthy. Its method of labeling entire files as healthy or sick instead of trying to label individual symptomatic sounds makes it much easier obtain labeled training data. The user can simply select files from time periods when it is known that the chickens were either sick or healthy, and does not need to tediously label each sound within the files.

However, the file-level labeling also comes with a drawback. It is more difficult to ensure that the algorithm is actually cuing on symptoms of the disease instead of on other potentially confounding factors, such as slight differences in the background noise for each isolator box. To verify whether or not this is the case, we would need additional datasets recorded in similar settings (e.g. with the isolator chambers reversed for the experiment and control groups).

4.3 Cough detection

We also did some earlier work attempting to detect coughs and sneezes from chickens sick with laryngotracheitis (LT). Although coughs are generally louder than rales, they also vary much more in how they sound and what they look like spectrally. Some coughs are very short and abrupt, while others last longer. Some are single bursts of sound ("choo") while others are double bursts ("ka-choo"). Many coughs are mixed with different vocalizations,

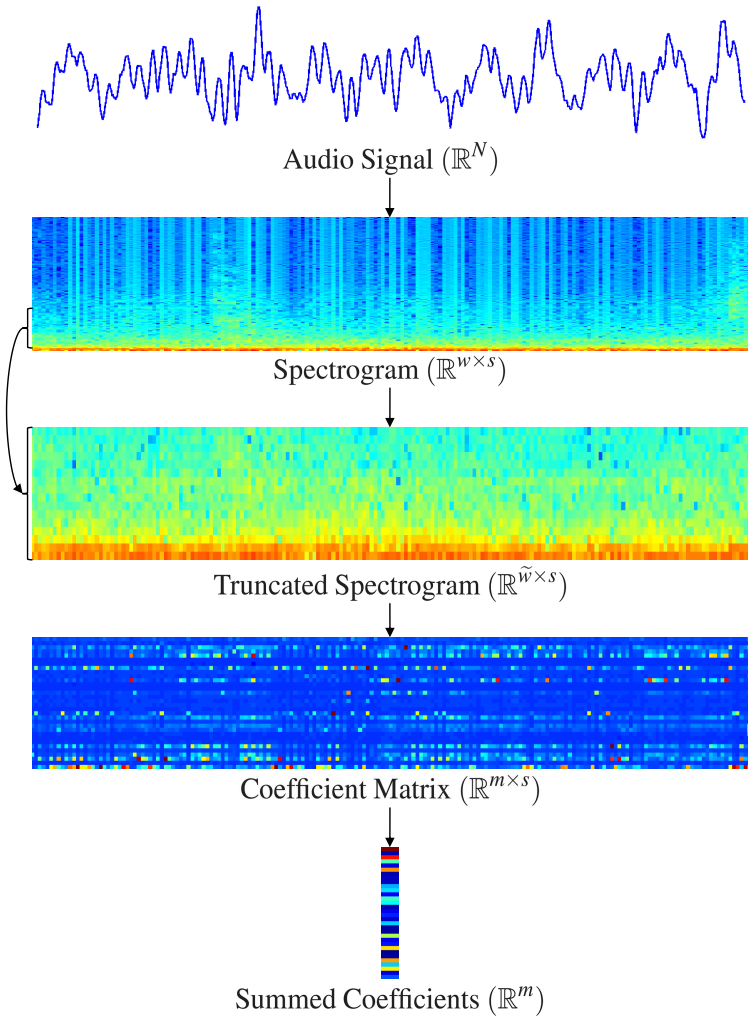

Figure 6: Feature calculation flowchart for IB detection. From the audio waveform, a spectrogram is computed. The frequencies between 500 Hz and 3 kHz are extracted from the spectrogram, and then decomposed into a sparse set of dictionary coefficients. These coefficients are summed together over the entire minute-long recording to yield the features used for classification. Figure taken from [108].

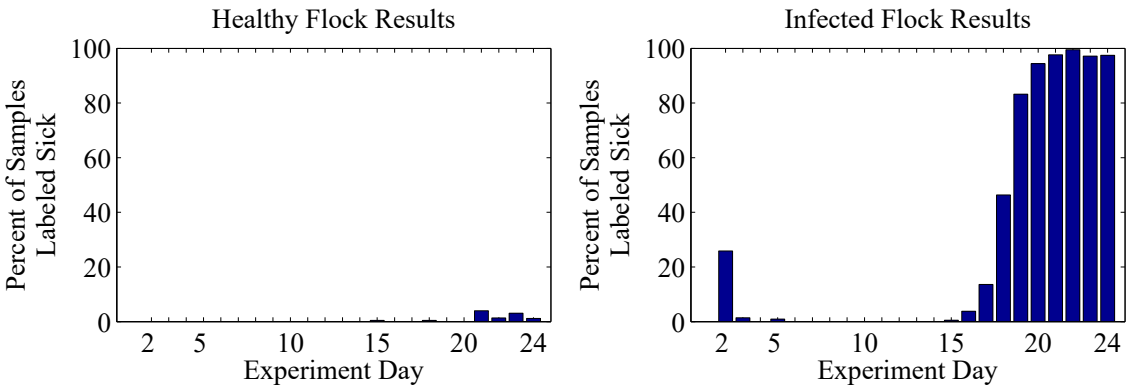

Figure 7: Daily percentage of IB audio recordings labeled as being sick for the control (*left*) and experiment (*right*) groups of chickens. The experiment group was infected on day 15. Figures taken from [108].

like chirps or a squawks, that change how they sound. We found that all of this variation made it difficult to separate cough sounds from other activity and noise that occurred during the day. However, the coughs were still present during the nighttime hours, while the background noise and activity levels were greatly reduced.

We developed a nighttime cough detection algorithm that was presented at the 2015 Poultry Science Association Annual Meeting [19]. In this algorithm, MFCCs were calculated from the audio data and then passed into an SVM trained to classify coughs. The SVM was trained using 20 minutes of labeled data from the LT2 dataset, and the resulting model was tested on the LT3 dataset. Figure 8 shows plots of our algorithm's nightly cough detection rate compared to plots of clinical sign scores measured by PDRC staff from each of the four rooms in the LT3 experiment. The clinical signs are a rough measure of how sick the birds were in each room, and were only measured over five days around the period when the birds should have been exhibiting symptoms of the disease. The birds in three of the four rooms were sick during this period, while the birds in the control room remained healthy. As can been seen in the plots, the number of coughs detected by our algorithm correlated well with the clinical sign measurements.

After classifying the individual feature samples, the number of cough detections was summed over a 60 minute wide sliding window and the result was compared against a threshold to classify the chickens as either healthy or sick. This threshold was swept across

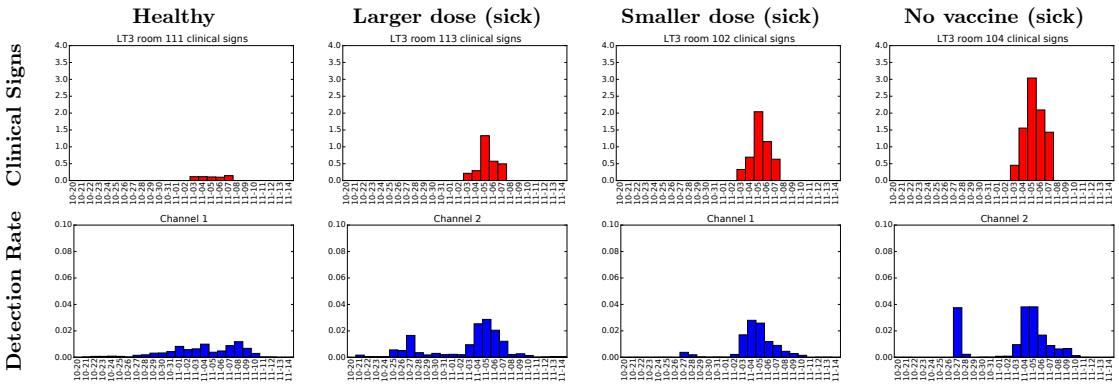

Figure 8: Plots of LT3 clinical signs measurements (*top*) versus the nightly cough detection rate (*bottom*). The clinical signs were only measured over the 5 days the birds were expected to be most sick. The left column shows the results for the room containing the control flock, which never got sick. The right three columns show the results for the three sick rooms. The early spike in the detection rate seen in the second and fourth columns occurred on the day the chickens were placed in the rooms.

the full range of values to generate receiver operating characteristic (ROC) curves showing the true positive rate from each of the three sick rooms plotted against the false positive rate from the healthy room. These ROC curves are plotted in Figure 9, and represent the nighttime data over the period of time when the chickens in the experiment rooms were actually sick. The results indicate that the algorithm can give about a 50% or greater true detection rate over a 60 minute window with almost no false positives.

The algorithm performed well on this data even though the healthy control group was located in different rooms between the training (LT2) and testing (LT3) datasets. However, the model trained here likely would not generalize well to other facilities without labeling new data and retraining.

4.4 Tools

In the process of testing different ideas and methods for disease detection, we developed software tools to make it easier to run trained models on different data and solicit help in labeling data.

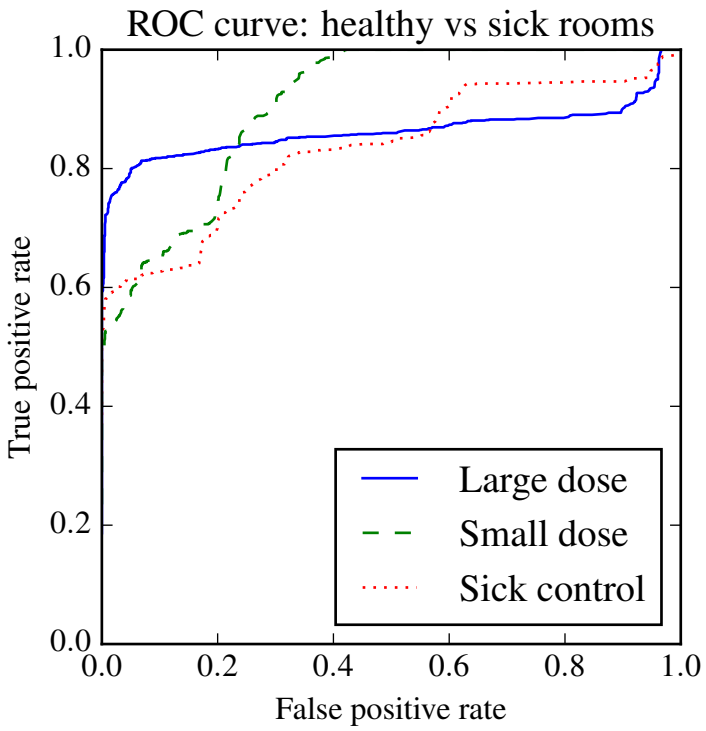

Figure 9: ROC curves for LT detection using nighttime LT3 data showing the true positive rate from each of the three sick rooms plotted against the false positive rate from the healthy room. Cough detections were summed over an hour long sliding window, and the results were compared against a threshold that was varied to generate the plot.

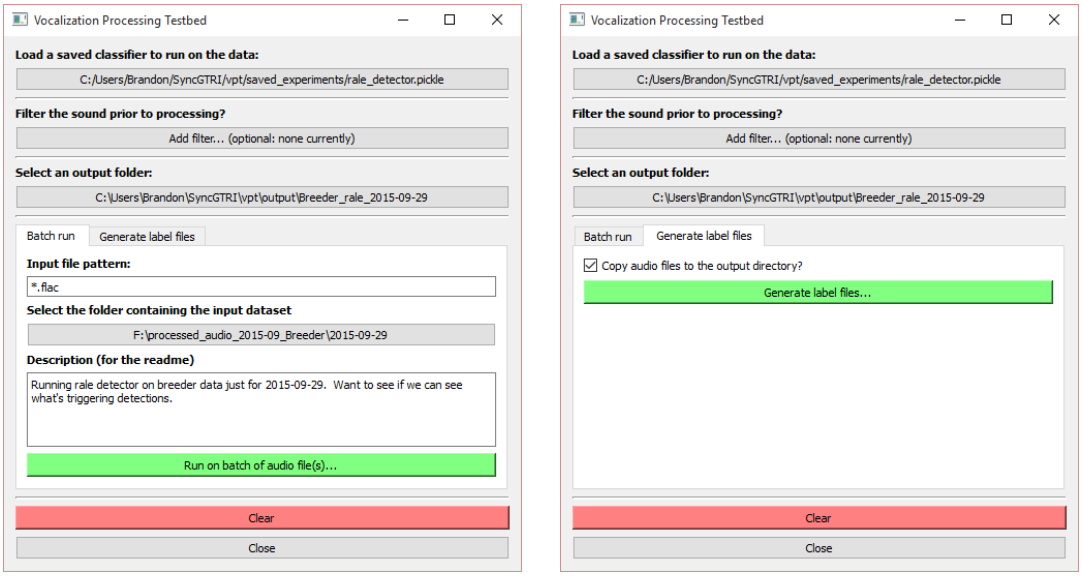

Figure 10: Vocalization processing testbed (VPT) screenshots. The left screenshot shows the tab that allowed the user to run a detector on the data in a folder and output the results to a comma-separated variable file. The right screenshot shows the tab that allowed the user to output label files that could be loaded in Audacity to see what sounds triggered detections.

4.4.1 Vocalization processing testbed (VPT)

The vocalization processing testbed (VPT) was a framework we developed in Python for our earlier disease detection algorithms. It defined an interface for each training or processing step that allowed them to be chained together easily. Complete, trained models could be saved out to a single file and loaded again for later use. The VPT included a GUI interface for loading and running these models on arbitrary data to allow them to be used by people unfamiliar with the code. The output could either be classification statistics for each of the input files, or label files that showed exactly what sounds triggered the different classifications when loaded in Audacity. Screenshots of the interface are given in Figure 10. We later moved away from Python and the VPT interface in order to make use of Rubinstein's optimized sparse coding algortihms, which were written in Matlab [96].

4.4.2 Labeling GUI

We also developed a labeling GUI in Matlab to make it easier and faster for animal science experts to help us label sounds in the data. The GUI required the sounds of interest in a

recording to be pre-segmented, allowing the user to quickly skip from sound to sound by clicking on the associated entry in a table. While this increased the preparatory work needed on our end, it decreased the effort required for those we would ask for help by eliminating the need to find and specify when the sounds occurred. The GUI allowed the user to play the context around each sound while animating the current play location on a spectrogram plot. It also included functionality to normalize the volume of the sounds since the gain settings sometimes varied from microphone to microphone.

CHAPTER 5

OUTLIER LEARNING VIA AUGMENTED FROZEN DICTIONARIES (OLAF)

A small-scale preliminary test of the method presented in this chapter was presented as an abstract in [20]. A much more comprehensive description and evaluation of the method was later published as a journal paper in [18]. Most of the contents of this chapter are taken from that paper.

This work was done in collaboration with Bradley Whitaker. He contributed the code dealing with the alternating minimization method and SVM learning, while I contributed the code dealing with K-SVD and feature calculation as well as the idea for the algorithm.

5.1 Introduction

People are adept at acclimatizing to an audio environment and identifying changes. Duplicating this behavior in software, however, is a difficult task and an active research area [6, 30, 45, 8]. This chapter presents outlier learning via augmented frozen dictionaries (OLAF), a signal processing method we have developed to detect anomalous events in an acoustic sound scene. First, typical sound events are learned for a given environment using a dictionary learning algorithm. The learned dictionary elements are then frozen (held constant), and the dictionary is augmented with additional elements and trained on data containing abnormalities. Since the original, frozen elements are still available to represent typical events, the new elements should learn to represent the anomalies that were not present in the normal data.

The OLAF method requires knowledge of the general time periods of when the data contains anomalies, but does not require labeling of the individual anomalous sounds. Its tendency to separate the anomalous and normal events can make subsequent classification of individual events easier and require less individually labeled training data. If individual events need not be detected, it also lends itself well to classifying anomalous periods of time

without the need for any individually labeled data.

The algorithm was tested on two sets of environmental recordings of chickens. One set contained anomalies that we synthetically added to some of the recordings, while the other contained a known anomalous condition where some of the chickens were sick for a period of time. The results with the sick chickens improve upon our prior disease detection work described in Sections 4.1 and 4.2 both in terms of performance and a reduced need for costly labeling of individual sound events. These types of algorithms have the potential to allow farmers to respond more quickly to the onset of disease, lowering their costs and the likelihood of diseases spreading to other houses.

Several aspects of these environments pose challenges for audio signal processing. The ventilation fans create significant amounts of background noise. Workers can create a wide variety of sounds and disturbances as they maintain equipment and care for the birds. The birds themselves make many different types of noises, including various vocalizations, flapping wings, drinker strikes, and feeding. The primary symptom of the disease the sick chickens had was that they would sometimes emit rales (subtle gurgling noises) while breathing. These rales are much quieter than many of the other sounds typically present.

A major benefit of the frozen dictionary approach is its generality. It could easily be applied to many other signal processing problems both inside and outside the field of bioacoustics—as long as data with and without anomalies is available.

5.2 Frozen elements algorithms

The frozen dictionary approach is a method we developed to try to learn the difference between normal and abnormal data in the presence of similar background noise. It is similar to work previously done by Smaragdis with non-negative matrix factorization [98, 99]. The method is as follows:

1. Learn a sparsifying dictionary on normal data.

2. Freeze the learned dictionary elements and augment the dictionary with additional non-frozen elements.

3. Run the dictionary learning algorithm again on the abnormal data, holding the frozen elements constant.

Ideally, the frozen part of the dictionary learned in the first step will be able to represent the background noise and normal aspects of the data well. When the learning algorithm is run again in the third step, the majority of the reconstruction error should come from the anomalous aspects of the data that were not present in the normal data. Thus, the augmented elements should learn to represent those abnormalities in order to minimize the reconstruction error.

We modified both the alternating minimization and the K-SVD algorithms mentioned in Sections 3.3.2.2 and 3.3.2.3 so that only a portion of the dictionary would be updated during each iteration.

5.2.1 Frozen K-SVD

Modification of the K-SVD algorithm to allow some dictionary elements to be held constant while others are updated is straightforward. The loop beginning on line 3 of Algorithm 2 in Section 3.3.2.3 is simply changed to only loop over the non-frozen columns of the dictionary.

5.2.2 Frozen alternating minimization

Modifying Algorithm 1 from Section 3.3.2.2 results in what we call the frozen alternating minimization algorithm, presented in Algorithm 3.

In our situation, the initial frozen dictionary $\mathbf{D}_f$ is the dictionary learned using Algorithm 1 on the dataset assumed to contain no anomalies. The initial unfrozen dictionary $\mathbf{D}_u$ is generated randomly. When the unfrozen dictionary is trained on data that contains additive anomalies, it should learn features that represent these anomalies since the normal part of the signal can already be modeled by the frozen dictionary.

5.3 Experiments

5.3.1 Data

The first set of recordings came from the Breeder dataset described in Section 2.1.2, consisting of a few hundred chickens in a long pen at a research farm. The second came from

Algorithm 3 The frozen alternating minimization algorithm, where each $\boldsymbol{y}_m$ is a data vector (a column of $\mathbf{Y}$), each $\boldsymbol{x}_m$ is a sparse coefficient vector (a column of $\mathbf{X}$), $\mathbf{D}$ is the sparse coding dictionary, λ is a fidelity-sparsity tradeoff parameter, $\mathcal{C}$ is the set of matrices in $\mathbb{R}^{\mathrm{N}\times\mathrm{K}}$ whose columns have ℓ^2-norm less than one, and M is the number of data samples selected for training. Algorithm adapted from [18].

Input: Signals $\{\boldsymbol{y}_m \in \mathbb{R}^{\mathrm{N}}\}_{m=1,...,\mathrm{M}}$, initial frozen dictionary $\mathbf{D}_f$, initial unfrozen dictionary $\mathbf{D}_u$, regularization term λ, number of iterations K

Initialize $\mathbf{D} \leftarrow [\mathbf{D}_f|\mathbf{D}_u]$
for $k = 1, ..., \mathrm{K}$ **do**
    for $m = 1, ..., \mathrm{M}$ (*in parallel*) **do**
        Calculate coefficient vectors:
        $\boldsymbol{x}_m = \arg\min_{\boldsymbol{x}} \frac{1}{2} \|\boldsymbol{y}_m - \mathbf{D}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$
    end for
    Update dictionary (unfrozen elements only):
    $\mathbf{D}_u = \arg\min_{\mathbf{D}_u \in \mathcal{C}} \frac{1}{\mathrm{M}} \sum_{m=1}^{\mathrm{M}} \frac{1}{2} \|\boldsymbol{y}_m - [\mathbf{D}_f|\mathbf{D}_u]\, \boldsymbol{x}_m\|_2^2$
    $\mathbf{D} = [\mathbf{D}_f|\mathbf{D}_u]$
end for
return D

the IB dataset detailed in Section 2.1.4, consisting of healthy and sick chickens in isolator boxes during a vaccine trial.

5.3.1.1 Chicken audio with synthetic anomalies

The training data for the synthetic anomaly experiment consisted of 420 minutes of data randomly selected from the three months of data in the Breeder dataset. Of these 420 minutes, 300 minutes were unmodified and served as the non-anomalous data. We synthetically added anomalies to the remaining 120 minutes of data.

The synthetic anomalies were random, 0.2 s to 0.4 s segments of a short recording of human crowd babble noise that were added to the waveform at a power level 6 dB below the power of the minute-long chicken recording. The power level of the anomalies was chosen to be low to prevent detection using simple signal-level techniques and also to simulate the detection of subtle events. This made the anomalies quiet enough that even human listeners would have difficulty noticing them.

The chicken recordings were resampled to 16 kHz to match the crowd babble recording prior to adding the anomalies. The insertion rate varied randomly between 10 and 30

anomalies per minute. The ends of the anomaly waveforms were tapered using a Hanning-shaped window, and the final waveform was scaled down if necessary to prevent clipping.

The above procedure was repeated to obtain another 300 normal and 120 anomalous one-minute-long files to serve as a test set, ensuring that no files overlapped with the training set.

5.3.1.2 Chicken audio with disease

For the IB data, we treated rale sounds produced by sick chickens as the anomalies. This sound is never present in the healthy control group, or for the experiment group prior to becoming sick. Our knowledge of the general time period when the experiment group was sick allowed us to apply the OLAF method.

We randomly selected 600 files from healthy chickens to serve as the normal training data. This included files from the control group throughout the entire experiment, and from the experimental group prior to infection. We randomly selected 120 files from the period when the experimental group was sick to serve as the anomalous training data. A second, disjoint set of 600 healthy and 120 sick files were selected as a test set. Finally, an additional 20 minutes of recordings from the experimental group were individually analyzed and the rales in those files were manually labeled.

5.3.2 Procedure

We used mel magnitude spectrum features (see Section 3.2.4) calculated as follows:

1. Apply a pre-emphasis filter (having a zero at 0.97) to the waveform.
2. Calculate the short-time Fourier transform (STFT) with a window width of 50ms and 50% overlap.
3. Take the magnitude of the result from the STFT.
4. Apply a triangular filter bank with 13 mel-scaled frequency bands between 100Hz and 8kHz to the magnitude spectrum.

This feature calculation is similar to that of mel-frequency cepstral coefficients (MFCCs) [40], but without the log scaling or the discrete cosine transform at the end. Leaving off these two steps preserves the pseudo-additivity of the magnitude spectrum, which matches the assumptions of additivity underlying the K-SVD algorithm. We also do not include an explicit energy term as is often done with MFCCs.

After extracting the audio features, we learned augmented frozen dictionaries using both the K-SVD and alternating minimization algorithms. When learning a dictionary using the alternating minimization algorithm, we applied a zero-mean preprocessing step on the audio features. When using the sparse coding algorithms, we learned dictionaries with and without the zero-mean operation. We did not do a full whitening by normalizing to unit variance. Normalizing the variance would increase the influence of noise, particularly at night when the chicken noises themselves are relatively quiet and the sound scene is dominated by background noise.

All dictionaries had 36 elements that were trained on the normal, or healthy, training data. Afterwards, we froze those 36 elements and augmented the dictionary with 4 additional elements. The augmented elements were trained using the unlabeled anomalous, or sick, training data. We used these dictionaries to calculate sparse coefficient vectors for the training and test data in each dataset. While the features are all non-negative, we do not enforce non-negativity when learning the dictionaries or calculating the coefficients.

The number of frozen elements and augmented elements was not chosen through a rigorous parameter search. These numbers were chosen somewhat arbitrarily, and future work could explore how sensitive the frozen dictionary approach is to these parameters. We chose the number of augmented elements to be small because there is little variance in the audio characteristics of chicken rales.

We used the learned coefficients as features in three different analyses. The first two analyses were applied to both the babble noise dataset and the hand-labeled portion of the IB dataset, and required that individual anomalies in the audio files be labeled. The third analysis was performed on the entire IB dataset, and only required the high-level labels needed to learn the frozen dictionary. The analyses are summarized as follows:

1. Average and compute the correlations of the coefficient magnitudes calculated using the normal data, the normal portions of the anomalous data, and the anomalous portions of the anomalous data.

2. Use the anomaly labels to train a support vector machine (SVM) classifying whether or not each 50 ms audio window contains an anomaly (either babble noise or a chicken rale, depending on the dataset).

3. Average the coefficients calculated across an entire one-minute audio file and classify whether or not each audio file came from the healthy flock or the sick flock.

5.4 Results

As previously mentioned, we use our sparse coefficients in two different types of anomaly detection. The first type is concerned with identifying individual anomalous events. While it may be tempting to classify an anomaly based on the augmented dictionary elements that are likely used to represent those anomalies, a more effective way to classify is by using common machine learning algorithms, such as SVMs. This type of anomaly detection requires that individual anomalies are labeled in the SVM training data.

The second type of anomaly detection identifies whether or not a given audio file comes from a normal dataset or an anomalous dataset. This method only requires the file-level labels that are used to generate the dictionaries.

In the remainder of this section, we analyze the coefficients by looking at how they are used in sparse-coding the test data. In addition, we use the coefficient vectors to train SVMs to determine whether or not a given 50 ms time window contains babble noise or a chicken rale according to the dataset. For the infectious bronchitis dataset, we also average the coefficient vectors from each one-minute audio file to train an SVM that can identify if a test file comes from a healthy flock or a sick flock. We use LIBSVM to train and test the SVMs [23].

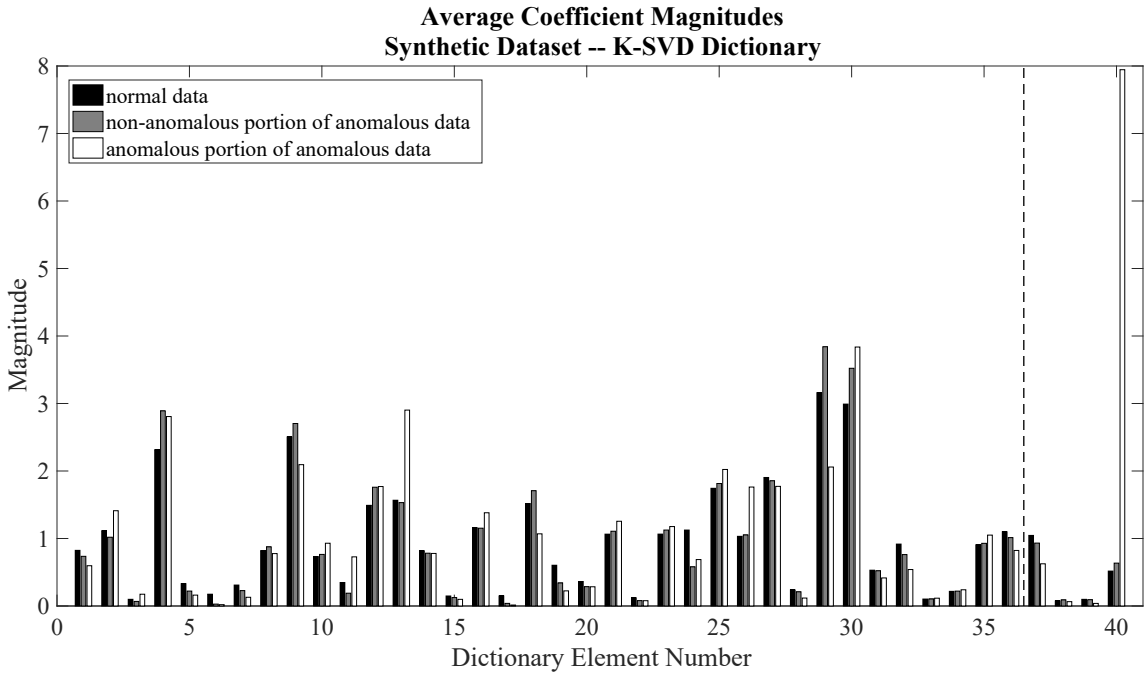

Figure 11: Average coefficient magnitudes of the synthetic anomaly test set using the K-SVD dictionary. Dictionary elements 1–36 were learned on the normal data and frozen while elements 37–40 were learned on the anomalous data. Figure taken from [18].

5.4.1 Synthetic anomalies results

5.4.1.1 Average coefficient use

The average coefficient magnitudes calculated using the test data from the synthetic anomaly dataset are plotted in Figures 11 and 12. Figure 11 shows the coefficients from the K-SVD dictionary without the zero-mean preprocessing step. The black bar in each group corresponds to the coefficients obtained from the non-anomalous data. The gray bar corresponds to the coefficients of the portions of the anomalous data that did not have anomalies. The white bar corresponds to the coefficients of the anomalous portion of the anomalous data. These bar graph conventions are the same in the other figures discussed in this section.

With regards to Figure 11, two observations are of particular importance. The first observation is that one of the augmented dictionary elements (element 40) is not used much when factoring the normal audio signal, but is used extensively on the anomalous portions of the anomalous data. This conforms to our hypothesis; after freezing the original 36-element dictionary, the augmented portion uncovered a dictionary element that seems to correspond

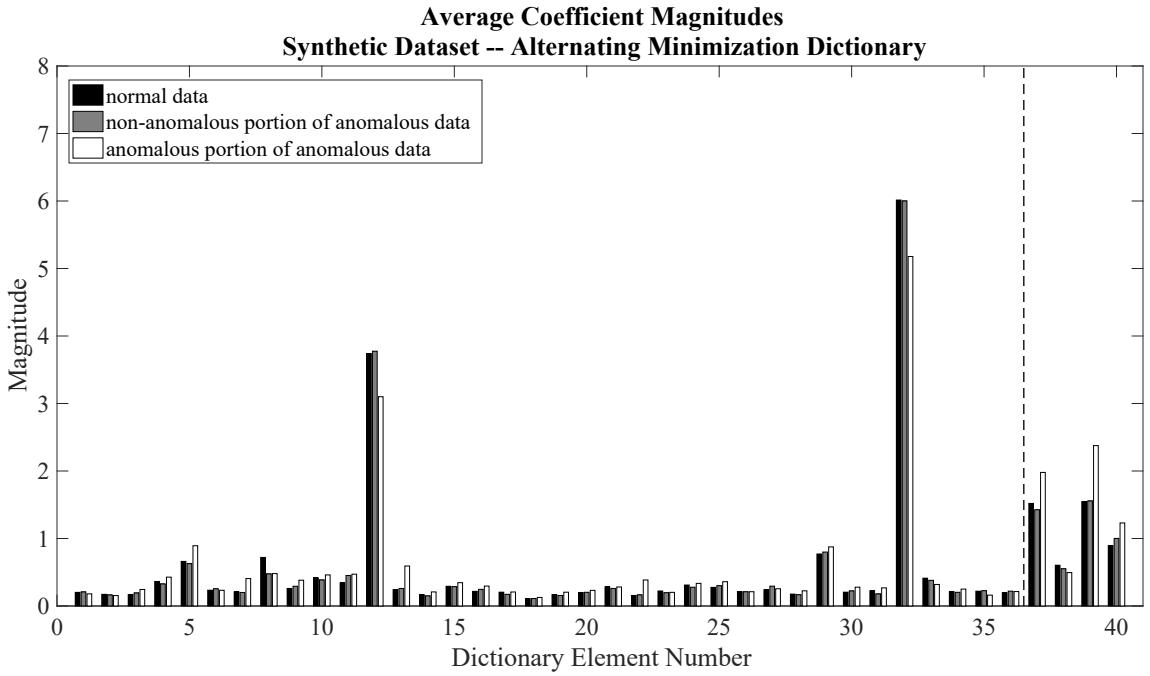

Figure 12: Average coefficient magnitudes of the synthetic anomaly test set using the alternating minimization dictionary. Dictionary elements 1–36 were learned on the normal data and frozen while elements 37–40 were learned on the anomalous data. Figure taken from [18].

to the audio features of the crowd babble noise.

The second observation is that the coefficients used by the dataset without anomalies and the normal portion of the dataset with anomalies are highly correlated. Indeed, the black and gray bars have a Pearson's correlation coefficient of 0.986. The black and white bars and the gray and white bars have much lower correlation coefficients of 0.502 and 0.525, respectively. This suggests that when calculating the sparse coefficients, different dictionary elements are used when modeling normal versus anomalous sounds. We note that while these (and subsequent) correlation values are not used as part of the classification process, we report them to validate the intuition of using the frozen dictionary method to learn discriminating features from the data.

Figure 12 shows a plot of the average coefficient magnitudes calculated using the alternating minimization dictionary. As with the K-SVD dictionary, the normal data and the normal portion of the anomalous data are almost perfectly correlated: the correlation coefficient is 0.999. However, the anomalous coefficient vectors are also highly correlated with

Table 3: Synthetic anomaly detection results. Results taken from [18].

	Accuracy ($\mu \pm \sigma$)	**Precision** ($\mu \pm \sigma$)	**Recall** ($\mu \pm \sigma$)
Unfiltered			
K-SVD	0.905 ± 0.003	0.643 ± 0.028	0.189 ± 0.032
K-SVD$-\mu$	0.911 ± 0.001	0.679 ± 0.018	0.272 ± 0.007
AM$-\mu$	0.922 ± 0.000	0.786 ± 0.008	0.268 ± 0.004
3-point median filter			
K-SVD	0.908 ± 0.003	0.869 ± 0.024	0.133 ± 0.035
K-SVD$-\mu$	0.917 ± 0.001	0.859 ± 0.025	0.236 ± 0.009
AM$-\mu$	0.924 ± 0.000	0.901 ± 0.011	0.243 ± 0.004

the normal coefficients: the black and white bars and the gray and white bars both have correlation coefficients of 0.976. Despite this, some elements (12, 13, 32, 37, and 39) still seem to distinguish between the babble noise and the regular audio. It is encouraging that two of these elements come from the augmented portion of the data.

5.4.1.2 SVM classifier

The synthetic anomaly dataset was completely labeled: we defined a 50 ms time-window as being anomalous if it contained at least 25 ms of crowd babble noise. With these labels, we were able to generate an SVM to test the effectiveness of using sparse coefficient vectors in a classification task. Table 3 outlines our results.

In this table—and in the other tables in this chapter—the "K-SVD" row reports the results of the SVM trained and tested on the coefficients calculated using the K-SVD dictionary that did not include the zero-mean preprocessing step; the "K-SVD$-\mu$" row reports the results for the K-SVD dictionary that included the zero-mean preprocessing operation; and the "AM$-\mu$" row reports the results for the alternating minimization dictionary (which also included the zero-mean operation as a preprocessing step). The tables report the mean and standard deviation of classification accuracy, precision, and recall after repeating the experiment ten times. In repeating the experiment, we used the same training and test data but used random initial conditions to learn new dictionaries and SVM classifiers. The small standard deviations reported in the tables indicate that the dictionary learning algorithms are very consistent.

The top half of Table 3 reports the accuracy, precision, and recall of the anomaly detection task with respect to individual 50 ms time windows.[1] The numbers represent the mean (μ) and standard deviation (σ) of scores after learning ten different dictionaries and their respective classifiers. The table shows that alternating minimization reports the highest average accuracy (0.922) and precision (0.786). The zero-mean K-SVD algorithm produces the best recall (0.272), but the alternating minimization algorithm's recall is very similar (0.268). All three algorithms are comparable, except that the K-SVD approach performed poorly in recall (0.189) and the alternating minimization approach performed much better in precision.

The bottom half of the table reports the same information after filtering the labels with a median filter. Recall that the synthetic anomalies range from 200 ms to 400 ms in duration. Since our features are calculated on 50 ms time windows, anomalies must occur in clusters; they cannot appear as isolated events. The three-point median filter will remove any anomaly that appears in an isolated time window. Applying the median filter increases the accuracy and precision in all cases, but does so at the expense of recall. Intuitively, this means that while we miss more anomalies, we are more confident that what we detect as an anomaly is classified correctly.

We note that while the recall is low for even the best-case scenario, the SVM is attempting to classify anomalies that have a power level that is 6 dB lower than the background audio sound. The SVM can still recall over a quarter of the almost inaudible anomalies.

We also wish to comment on the small recall values associated with the K-SVD case. A small recall indicates that the classifier had more false negatives. Since there is a large imbalance in the class sizes, having many false negatives (labeling many abnormal samples as normal) can still result in a high accuracy even though the classifier is not as useful. Because of this, the precision and recall values are often more informative when dealing with imbalanced datasets.

[1] Precision = TP / (TP+FP), recall = TP / (TP+FN), and accuracy = (TP + TN) / (TP + TN + FP + FN) where TP, FP, TN, and FN are the number of true positives, false positives, true negatives and false negatives, respectively.

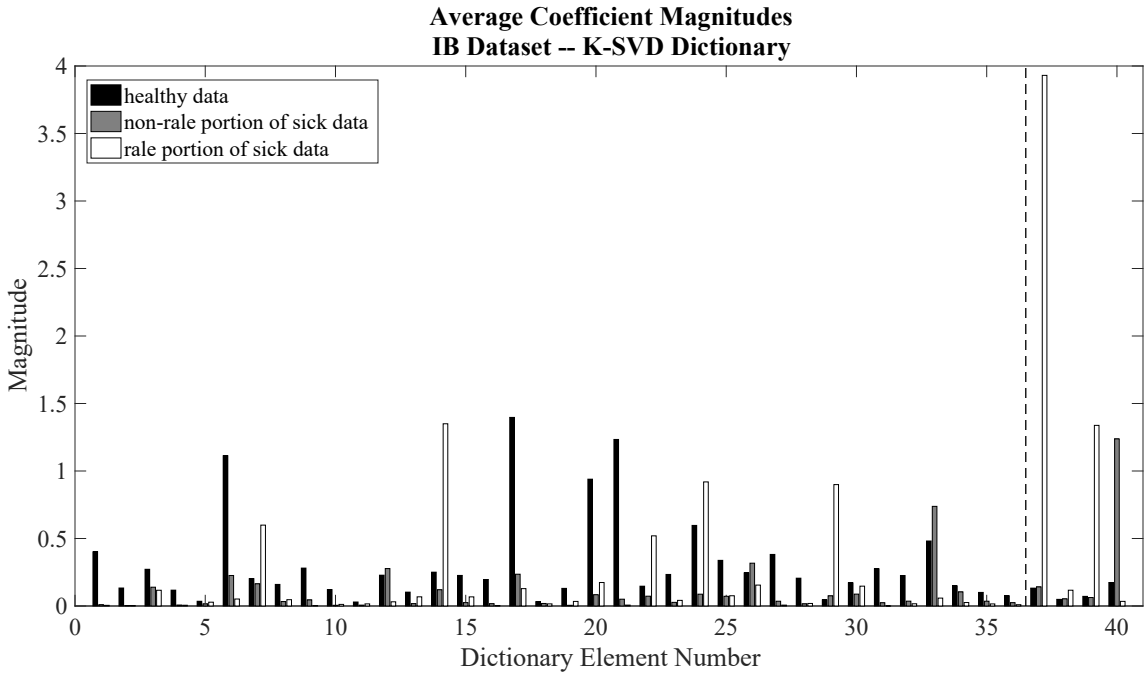

Figure 13: Average coefficient magnitudes of the infectious bronchitis test set using the K-SVD dictionary. Dictionary elements 1–36 were learned on the healthy data and frozen while elements 37–40 were learned on the sick data.

5.4.2 Disease detection results: individual time windows

5.4.2.1 Average coefficient use

The average coefficient magnitudes calculated using the test data from the infectious bronchitis dataset are plotted in Figures 13 and 14. Figure 13 shows the coefficients from the K-SVD dictionary without the zero-mean preprocessing step. We note that two of the augmented dictionary elements (37 and 39) are used heavily to reconstruct time-windows with rales present. We also note that, unlike in the controlled synthetic anomaly dataset, there is little correlation between any of the average coefficient values. The correlation coefficients between the black and gray bars, the black and white bars, and the gray and white bars are 0.1309, -0.0989, and 0.0035, respectively. The zero-mean K-SVD dictionary reports similar results.

One possible explanation of the poor correlation between the healthy test set and the non-rale portion of the sick test set is that while the non-labeled healthy test set came from a uniform sampling of the chickens, the labeled rale test set consisted of 20 one-minute-long

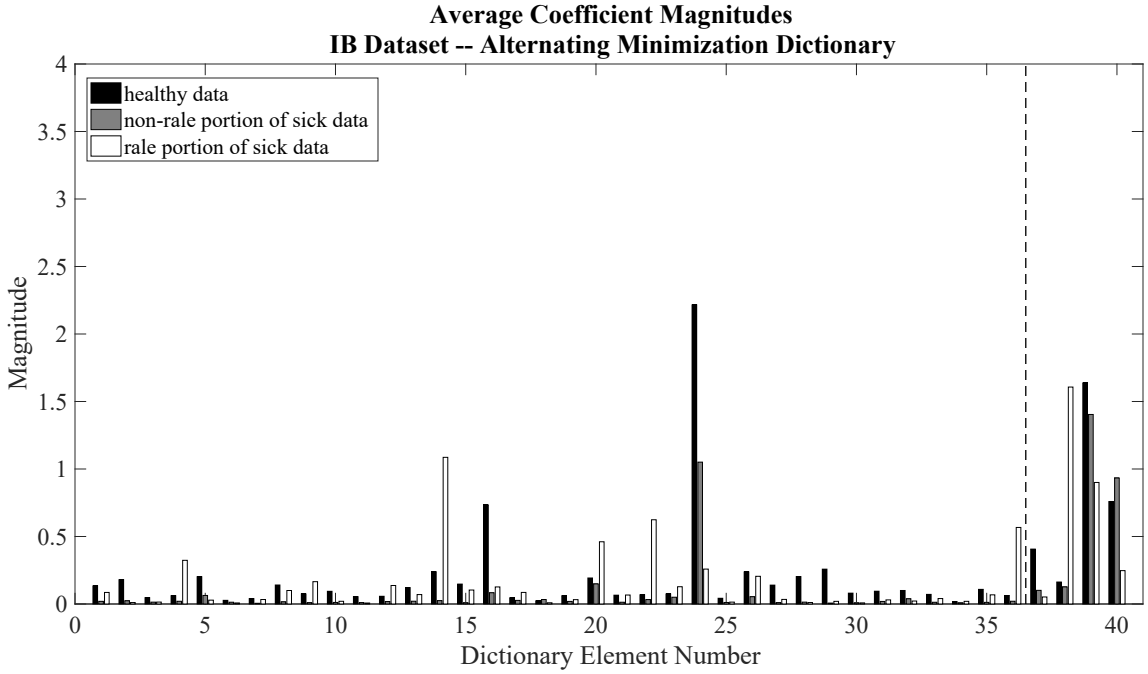

Figure 14: Average coefficient magnitudes of the infectious bronchitis test set using the alternating minimization dictionary. Dictionary elements 1–36 were learned on the healthy data and frozen while elements 37–40 were learned on the sick data.

files that were mostly recorded at night. This is because rales are more easily identified and labeled while the chickens are asleep and background noise is minimized.

Figure 14 shows the coefficients calculated using the alternating minimization dictionary. As in the previous figures, one augmented dictionary element stands out as correlating particularly well with the rale audio sounds (element 38). While it may not appear so at first glance, these average coefficient magnitudes also have desirable correlation properties. The healthy dataset and the non-rale portion of the sick dataset have a correlation of 0.895, the healthy dataset and the rale portion of the sick dataset have a correlation of 0.246, and the non-rale and rale portions of the sick dataset have a correlation of 0.325.

5.4.2.2 SVM classifier

Unlike with the synthetic anomaly dataset, we only had a small selection of manually-labeled data in the infectious bronchitis dataset. Therefore, we chose to train a 10-fold cross-validated SVM to determine how well the sparse coefficients could be used to classify individual time windows as containing a rale. Table 4 outlines our results.

Table 4: Individual rale detection results. Results taken from [18].

	Accuracy $(\mu \pm \sigma)$	**Precision** $(\mu \pm \sigma)$	**Recall** $(\mu \pm \sigma)$
Unfiltered			
K-SVD	0.968 ± 0.001	0.881 ± 0.017	0.791 ± 0.017
K-SVD$-\mu$	0.968 ± 0.001	0.875 ± 0.021	0.791 ± 0.018
AM$-\mu$	0.969 ± 0.000	0.862 ± 0.005	0.763 ± 0.004
3-point median filter			
K-SVD	0.963 ± 0.001	0.914 ± 0.014	0.701 ± 0.025
K-SVD$-\mu$	0.963 ± 0.001	0.912 ± 0.018	0.700 ± 0.029
AM$-\mu$	0.963 ± 0.000	0.911 ± 0.005	0.632 ± 0.006

The top half of the table reports the results for the unfiltered labeled rales. All methods have similar accuracy, precision, and recall values. In addition, these values are significantly higher than prior results of 73.4% precision and 51.4% recall reported in Section 4.1. The increase in performance can likely be attributed to a number of factors. The clustering approach used in the prior work cannot effectively represent multiple sounds happening at once without creating new clusters to represent each different possible combination of sounds. The dictionary approach used here can efficiently represent combinations of sounds as the sum of a few underlying elements. The previous work also had fewer files with labeled rales to use for training, and the decision tree algorithm used for its classifier is likely not as robust as an SVM.

The bottom half of the table reports the results after filtering the labels with a median filter. The intuition behind the median filter is the same as with the synthetic anomalies: rales are not temporally isolated events, but are usually continuous noises over 50–400 ms. As was true for the synthetic anomaly dataset, applying the median filter increased the precision of the classifier but reduced the recall.

5.4.3 Disease detection results: minute average

Manually labeling individual rales in audio data requires experts and is difficult and time-consuming. However, labeling whether a one-minute audio file came from a healthy flock or a sick flock is simple. To leverage this simplicity, we averaged the sparse coefficients across each one-minute file and labeled the entire file as healthy or sick. We learned an SVM on

Table 5: Minute-average IB detection results. Results taken from [18].

	Accuracy ($\mu \pm \sigma$)	**Precision** ($\mu \pm \sigma$)	**Recall** ($\mu \pm \sigma$)
K-SVD	0.985 ± 0.004	0.946 ± 0.024	0.971 ± 0.015
K-SVD$-\mu$	0.997 ± 0.002	0.990 ± 0.004	0.994 ± 0.008
AM$-\mu$	0.993 ± 0.003	0.977 ± 0.014	0.983 ± 0.011

the average coefficients, and the classifier results are reported in Table 5. The alternating minimization and zero-mean classifiers have a higher accuracy than previous work that also classified audio files using features that were averaged across one-minute files. Previous work reports an accuracy of 97.85% [108], but does not report precision or recall. In addition, the zero-mean K-SVD method presented in this paper achieves almost perfect results; on average, it correctly classifies 599 of 600 healthy files and 119 of 120 sick files.

5.5 Conclusions

Our frozen dictionary approach to anomaly detection can be used to detect the presence of subtle events in acoustic sequences without the need to label those events specifically. Using this approach, it would be easy to create detectors for specific, natural events by first training on background sounds and then presenting samples that contain both the background and the desired sound. If labels are available, our approach can be used to create features and a classifier that can be used to detect individual events.

CHAPTER 6

ESTIMATING THE LIKELIHOOD OF SPARSE APPROXIMATIONS (ELSA)

6.1 Introduction

In many applications, it is interesting to know when a system deviates in some way from its normal mode of operation. Usually, data from the system operating under normal conditions is readily available, but data from the system under anomalous conditions is not. There are often innumerable and unpredictable ways in which a system can behave in an anomalous fashion, making collection of data under every possible anomalous condition impossible. Even for known anomalous modes, it may be too costly or otherwise infeasible to create the anomalous conditions just to collect data. For example, no chicken farmer would intentionally infect an entire flock just to collect disease data from a particular facility, and neither would it be ethical to do so.

In this chapter, we present a method to estimate the likelihood of sparse approximations (ELSA) for test data in order to quantify how much a system has deviated from its normal behavior. Unlike the OLAF method presented in Chapter 5, ELSA does not require any training data from the anomalous modes of operation—although it does still require data from normal conditions. Since normal data is generally easy to obtain, ELSA is easier to adapt and apply to new situations, whereas OLAF may be better for targeting a known mode of anomalous behavior for which data is available.

6.2 Prior work

The ELSA method fits best in the area of novelty detection, which involves the task of recognizing when there has been a change in the underlying processes that generate observed data. Pimentel et al. reviewed the area of novelty detection in 2014 [90], and other reviews have covered the closely related areas of anomaly detection [22] and outlier detection [112]. However, none of these reviews mention any work based on sparse representations, as is

proposed here.

Although not mentioned in the reviews, some novelty detection work based on sparse representations has been done in the past few years. Most of this work has dealt with detecting anomalies in images [111, 10] and video [110, 67, 37, 78], although there is also a paper dealing with ECG data [1]. All of these methods examine metrics based on the reconstruction error and sparsity to determine when anomalies occur. The ELSA method incorporates the reconstruction error in its monitoring, but is unique in how it also monitors which atoms get used and how well their coefficient values fit the expected distributions.

The ELSA method represents a mixture of two of the categories of novelty detection methods that Pimentel enumerates in his review [90]. The sparse coding component fits in with the reconstruction-based methods, and the subsequent estimation of probability distributions over the sparse coefficent values fits in with the probabilistic techniques.

6.3 The ELSA method

The ELSA method characterizes a given condition within an environment in two steps. First, it learns the structure of the various different types of sounds that are present in the environment. Second, it learns a model representing the characteristics of those different types of sounds under the given condition. Both steps require data collected from the relevant environment or condition for training. However, the same data can be used for both steps—particularly when the goal is to measure deviation from normal behavior. When characterizing multiple different conditions within an environment, the results of the first step can be shared across all the models for the different conditions.

The two steps above each produce a set of information used in evaluating the similarity of future data to the training data. The first produces a sparse coding dictionary that attempts to capture the general structure of the data (e.g. the types of sounds that are present). The second produces a set of probability distributions estimated over the training data that characterizes the likelihoods of each dictionary atom being used, of the non-zero coefficient values associated with each atom, and of the magnitude of the residual error from the sparse representation. With this information, we can decompose new data and estimate

its likelihood of occurring under the assumption that the system is operating under the same circumstances from which the training data was drawn. When that likelihood is small, it indicates that the assumption is incorrect and the system is operating under different conditions. The process can be summarized as follows:

1. Learn a sparse coding dictionary $\mathbf{D}$ on a set of data $\mathbf{Y}_\mathrm{d}$.

2. Use the dictionary $\mathbf{D}$ to perform sparse decomposition on a set of training data $\mathbf{Y}_\mathrm{t}$ representative of the mode $\mathcal{M}$ to be characterized, producing a sparse coefficient matrix $\mathbf{X}_\mathrm{t}$ such that $\mathbf{Y}_\mathrm{t} \approx \mathbf{D}\mathbf{X}_\mathrm{t}$.

3. Estimate over the training data $\mathbf{Y}_\mathrm{t}$ the probabilities of each atom getting used, the probability density functions (PDFs) for the non-zero coefficient values in $\mathbf{X}_\mathrm{t}$ for each atom, and the PDF for the magnitude r of the residual error for each sample.

4. Sparse code the test data $\mathbf{Y}$ as $\mathbf{Y} \approx \mathbf{D}\mathbf{X}$, then estimate the log-likelihood of each column in $\mathbf{Y}$ using the log-likelihoods of each atom used, the log-likelihoods of the non-zero coefficient values from the corresponding column of $\mathbf{X}$, and the log-likelihood of the magnitude r of the residual error.

5. Compare the log-likelihoods of the test data against the log-likelihoods typically seen under mode $\mathcal{M}$ or under other modes of operation to make a classification or decision.

Each of these steps is detailed in the following sections.

6.4 Dictionary learning (step 1)

The first step of the ELSA method is to learn a dictionary that can be used to construct sparse representations of the data. Since many sounds may be present in many of the different modes of operation that we hope to characterize in a chicken house, we generally prefer to learn a single dictionary over a broad swath of data. This allows, for instance, sounds such as chirps to be represented by the same dictionary elements whether they occur during the day or during the night. Ideally, the dictionary will be able to represent all the types of sound present in the environment, while their frequency and loudness will later be characterized by the estimated probability distributions.

Another approach could be to train the dictionaries on a narrower set of data that targets a specific mode or condition. In this case, the dictionary may not be able to represent well sounds that were not present in the training data. This could cause the dictionary atoms and their coefficients to become less informative in determining whether or not that condition is present, while the residual error would become more informative.

To learn a dictionary $\mathbf{D}$ (and its corresponding sparse coefficient matrix $\mathbf{X}_\mathrm{d}$), we would like to solve

$$\mathbf{D}, \mathbf{X}_\mathrm{d} = \underset{\mathbf{D}, \mathbf{X}_\mathrm{d}}{\arg\min} \|\mathbf{Y}_\mathrm{d} - \mathbf{D}\mathbf{X}_\mathrm{d}\|_\mathrm{F}^2 \qquad \text{subject to} \qquad \|(\boldsymbol{x}_\mathrm{d})_i\|_0 \leq s \quad \forall i, \tag{7}$$

where $\mathbf{Y}_\mathrm{d}$ is the data over which the dictionary will be trained, $\mathbf{X}_\mathrm{d}$ is the sparse coefficient matrix for that data, $(\boldsymbol{x}_\mathrm{d})_i$ is the ith column of $\mathbf{X}_\mathrm{d}$, and s is the sparsity constraint (the maximum number of dictionary atoms that can be used to represent a sample). Since the optimal solution to Equation (7) cannot be found efficiently, we instead get a fast approximation of the solution using the K-SVD algorithm. This step is depicted in Figure 15, and the K-SVD algorithm is covered in more detail in Section 3.3.2.3. Note that other dictionary learning algorithms for sparse representations that minimize different objective functions may also be used for this step.

6.5 Sparse decomposition of a particular mode (step 2)

Once a dictionary has been obtained, the orthogonal matching pursuit algorithm can be used to perform sparse decomposition of data representative of a particular mode (or condition) of operation $\mathcal{M}$, giving a sparse coefficient matrix (see Figure 16). The goal of the ELSA algorithm is to measure how different other data is from the data selected here. As such, there are a few different approaches to choosing which data to characterize, how to apply the ELSA algorithm, and how to interpret its output. These choices should be made in the context of the application, the user's objectives, and what information is available about the data.

The simplest approach is to broadly choose data representing normal operation and use the ELSA method to detect deviations from that baseline. This method works best when the normal data is fairly consistent, but can still give interesting results even when the normal

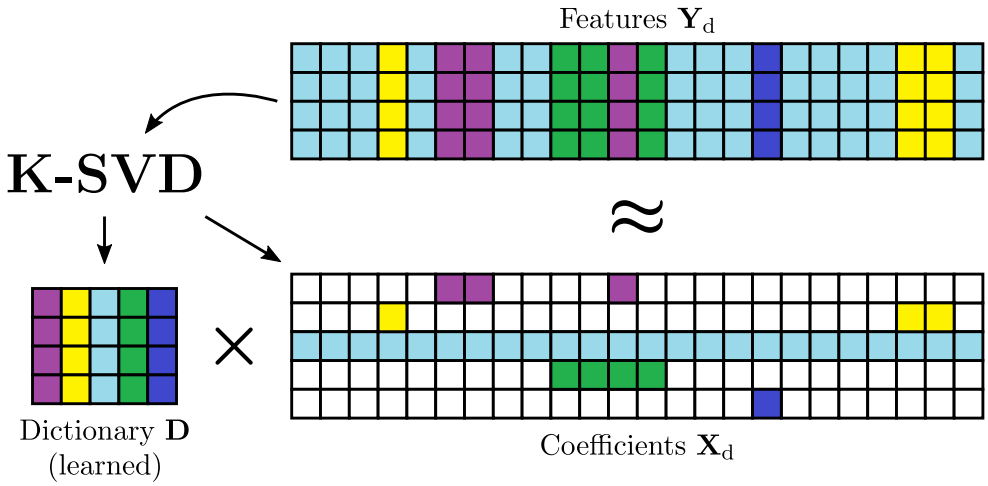

Figure 15: Dictionary learning (illustration). Training data $\mathbf{Y}_\mathrm{d}$ is fed into the K-SVD algorithm, which outputs a learned dictionary $\mathbf{D}$ and a sparse coefficient matrix $\mathbf{X}_\mathrm{d}$ such that $\mathbf{Y}_\mathrm{d} \approx \mathbf{D}\mathbf{X}_\mathrm{d}$. In the illustration, each column of the $\mathbf{Y}_\mathrm{d}$ matrix represents a vector of features calculated over a window of time, with different colors representing different types of sounds present in the data. The dictionary matrix ideally learns an atom (or column) matching each of the different types of sound present in the features. The white squares in the coefficient matrix represent zeros, while the colored squares represent non-zero coefficients and are colored to match their corresponding dictionary atom. In the illustration, the light blue squares could represent a continuously running fan, while the other colors could represent shorter sounds such as chirps.

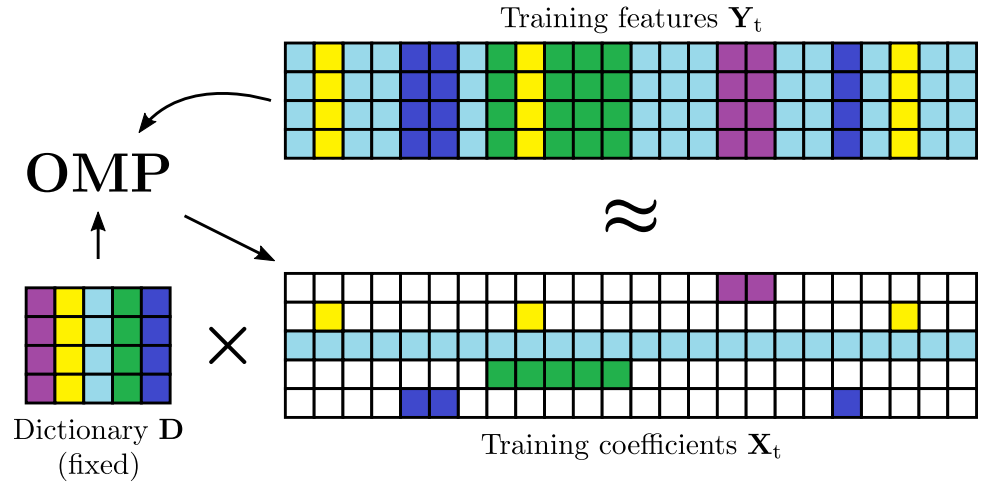

Figure 16: Sparse coding of audio features (illustration). The learned dictionary $\mathbf{D}$ and the training data $\mathbf{Y}_\mathrm{t}$ selected for mode $\mathcal{M}$ are passed into the OMP algorithm, which outputs a sparse coefficient matrix $\mathbf{X}_\mathrm{t}$ such that $\mathbf{Y}_\mathrm{t} \approx \mathbf{D}\mathbf{X}_\mathrm{t}$. The same algorithm and dictionary are later used to generate the coefficients $\mathbf{X}$ for the test data $\mathbf{Y}$.

data is strongly multi-modal. Examination of the log-likelihood plots produced this way can often help the user find modes or areas of interest within the data. For unlabeled poultry data, we have often used this method to find when the lights turn on and off by looking for step changes in the log-likelihood plots.

If there are known modes in the normal behavior of the system, data can be selected from each mode to build separate ELSA models. These models can then be run on new data and the outputs compared to determine what the most likely mode of operation is. If none of the models match well against the data, this can be an indication that an anomaly has occurred. For poultry data, we generally train separate models for when the lights are on and when the lights are off, as there are significant acoustic differences between when the chickens are active versus when they are sleeping. The outputs of these two models are often helpful in finding other modes of interest in the data.

The ELSA approach can also be flipped around if data from an anomalous mode is available. An ELSA model can be trained on the anomalous data, and then the anomaly can be declared if that model matches future data well. Similar to above, one or more models for anomalous modes could also be included in the set of models run against the data to get a classification of those anomalous modes.

6.6 Estimating probability distributions (step 3)

To estimate the likelihood of the test data $\mathbf{Y}$ being generated under mode $\mathcal{M}$, we first need to develop a probabilistic model of $\mathcal{M}$ based on the sparse representation of its training data. To do this, we'll look at both the sparse coefficients $\mathbf{X}_t$ and the residual error $\mathbf{E}_t = \mathbf{Y}_t - \mathbf{D}\mathbf{X}_t$ from the training data. Ideally, the coefficients will give us an idea of what structure is present that we have previously seen in the training data, while the error will give us an idea of how well the sparse model is able to represent the data.

With a dictionary containing q atoms, there are q coefficient values for each sample in the sparse representation. However, the sparsity constraint s only allows s of them to be non-zero for any given sample, which causes strong dependencies among the coefficient values. To get a parametrization that more closely matches the actual number of free parameters,

we represent each data sample as a list of indices $\{a_i\}_{i=1}^{s}$ of the atoms used to represent that sample and a list of the non-zero coefficient values $\{c_{a_i}\}_{i=1}^{s}$ associated with each of those atoms. While there are still dependencies among these parameters, they are generally much more independent from each other than the full list of coefficient values would be.

We include the residual error in our model as a way to handle samples that cannot be represented well using the trained dictionary. As such, we are primarily interested in the magnitude $r = \|\boldsymbol{e}\|_2$ of the residual error vector $\boldsymbol{e}$ for the given sample as opposed to the actual error along each of the different feature dimensions.

With the above parametrization, we can write the likelihood density for a given test sample $\boldsymbol{y}$ as

$$\mathcal{L}(\mathcal{M}|\boldsymbol{y}) = \mathrm{p}(\boldsymbol{y}|\mathcal{M}) = \mathrm{p}(a_1, a_2, \ldots, a_s, c_{a_1}, c_{a_2}, \ldots, c_{a_s}, r|\mathcal{M})\,. \tag{8}$$

To make estimation and computation of this likelihood tractable, we make the strong independence assumption common to naive Bayes methods and treat it as the product of independent distributions,

$$\mathcal{L}(\mathcal{M}|\boldsymbol{y}) = \mathrm{p}(r|\mathcal{M}) \prod_{i=1}^{s} \mathrm{p}(a_i|\mathcal{M})\, \mathrm{p}(c_{a_i}|\mathcal{M})\,. \tag{9}$$

The following sections detail how we use the sparse coefficients of the training data $\mathbf{X}_\mathrm{t}$ to estimate models of $\mathrm{p}(a_i|\mathcal{M})$ and $\mathrm{p}(c_{a_i}|\mathcal{M})$ for each atom, as well as how we estimate a model of $\mathrm{p}(r|\mathcal{M})$.

6.6.1 Atom usage probability

The atom usage probabilities are the easiest to estimate because they are discrete. First, a matrix $\mathbf{I}_\mathrm{t}$ indicating which atoms are used for each sample is obtained by binarizing the coefficient matrix $\mathbf{X}_\mathrm{t}$ such that any non-zero entry turns into a 1:

$$(\mathbf{I}_\mathrm{t})_{i,j} = \begin{cases} 0 & \text{if } (\mathbf{X}_\mathrm{t})_{i,j} = 0 \\ 1 & \text{otherwise} \end{cases} \qquad \forall i, j. \tag{10}$$

Recalling that a_i denotes the index of the ith atom chosen to be used in the sparse representation for a given sample, we can estimate the probability that atom a_i will be used given

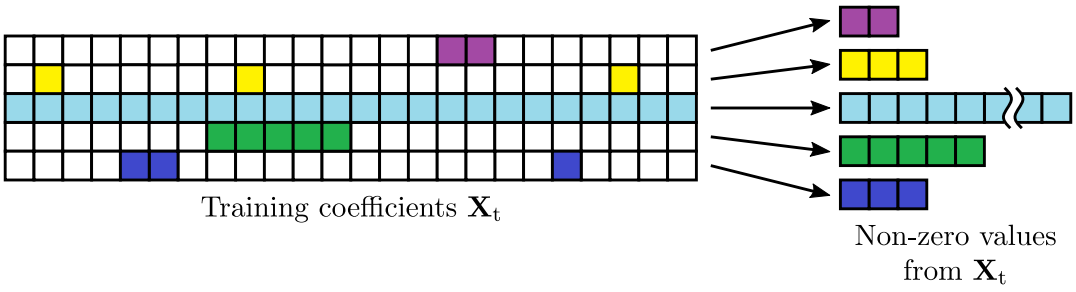

Figure 17: Extraction of non-zero coefficient values (illustration). After a sparse coefficient matrix is obtained for the data to be characterized, the coefficients for each atom (the rows) are broken apart and non-zero values (represented by white squares) are discarded.

mode $\mathcal{M}$ by finding the mean of the a_ith row of $\mathbf{I}_\mathrm{t}$. If there are n_t samples in the training data $\mathbf{Y}_\mathrm{t}$, this can be written as

$$\mathrm{p}(a_i|\mathcal{M}) = \frac{1}{n_\mathrm{t}} \sum_{j=1}^{n_\mathrm{t}} (\mathbf{I}_\mathrm{t})_{a_i,j}. \tag{11}$$

6.6.2 Coefficient PDFs

To characterize the coefficient values for each dictionary atom, we would like to estimate probability density functions (PDFs) for each atom's non-zero coefficient values in $\mathbf{X}_\mathrm{t}$. This will allow us to estimate the probability density $\mathrm{p}(c_{a_i}|\mathcal{M})$ for the value of the ith non-zero coefficient for a sample, given mode $\mathcal{M}$. Figures 17 and 18 provide conceptual illustrations of the extraction of non-zero values from $\mathbf{X}_\mathrm{t}$ and the estimation of PDFs from those values, respectively.

By considering only the non-zero coefficient values, we allow the system to learn separate PDFs for each of the different sound structures contained in the learned dictionary. When the sparse coding algorithm chooses which dictionary atoms to use to best represent a given sample, it also determines which PDFs should be used by ELSA to estimate the likelihood of that sample. This allows a kind of mode-switching within ELSA based on what structure is present in a given sample, enabling ELSA to better handle multi-modal data.

For instance, chirping sounds in chicken houses are common under normal conditions, but may be relatively rare compared to constant fan noise. A large, joint distribution would average the chirp sounds in with the much more prevalent fan noise, causing chirping to look abnormal. Under ELSA, the chirping and fan sounds would (ideally) be represented by

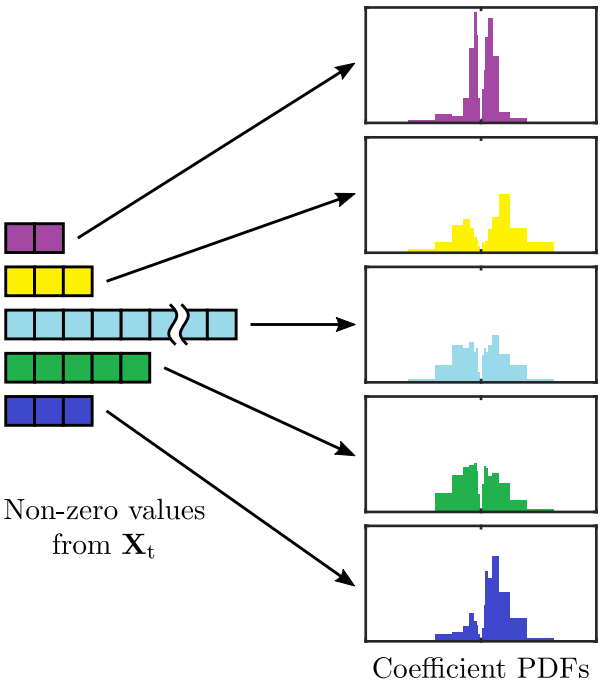

Figure 18: Estimation of PDFs from non-zero coefficient values (illustration). The PDF for each dictionary atom is estimated by taking a histogram over its non-zero coefficient values. The bins of the histograms are exponentially spaced so that they work well for various different circumstances and microphone gain settings. The distributions generally have a dip immediately around zero since zero values are not included and other atoms are more likely to be chosen if the coefficient value for the atom under consideration would be close to zero.

different dictionary atoms, each with their own PDFs estimated only over the times when those sounds were present. Since the frequency of the occurrence of the sound is ignored here, both the fan and the chirping noises could look normal under ELSA's coefficient PDFs. Information about the frequency of the occurrence of the sound is separated out into the atom usage probabilities from Section 6.6.1.

There are a number of approaches that could be used to estimate the PDFs—most notably, kernel density estimation (KDE, also known as the Parzen-Rosenblatt window method). However, in this work we use a simple histogram estimator. Although KDE would likely provide a higher fidelity estimate of the true PDF and would avoid binning problems, those benefits are outweighed by its drawbacks for our application. Since KDE combines copies of a kernel function centered on each training sample, its storage and computation requirements increase linearly with the number of training samples available. We often use training sets large enough to make this impractical. KDE would also increase parameter selection complexity because of the need to choose the type of kernel function as well as any parameters associated with that kernel function.

With a histogram estimator, we only need to store a single number for each bin regardless of the amount of training data used. Also, computing probabilities for new data only requires looking up values from the correct bins. The representation will not be as smooth as what KDE would provide, but this is mitigated by the fact that we usually aggregate many different probabilities together before making a decision.

The primary difficulty in estimating a PDF with a histogram is choosing the locations of the bin edges appropriately. Bins that are too wide may obscure some of the shape of the underlying distribution, while bins that are too thin will begin to capture the shape of the noise. We reasoned that when coefficient values are generally small, smaller deviations are more important and need correspondingly smaller bins to capture them. Similarly, larger coefficient values need larger deviations to be important. Thus, we decided use a logarithmic spacing for the boundaries between histogram bins. Specifically, we placed 10 bin edges per decade between -10^4 and -10^{-4}, one edge at zero, and 10 edges per decade between 10^{-4} and 10^4. This provides a wide enough range to cover the smallest (non-zero) and largest

coefficient magnitudes we've typically seen across all of our datasets.

Empirically, we have found that the exponentially spaced bins are able to represent the underlying distributions well despite using the same bin settings across datasets with drastically different environments and microphone gain settings. This is demonstrated by Figure 19, which shows example PDF estimates from three different datasets. Figure 20 shows the same PDF estimates, but using a log-scaled x-axis so that all the bins are clearly visible.

Another potential issue with using a histogram to estimate the PDF is that there are often bins in which no training samples fall. The value of the histogram for these bins will be exactly zero, which can cause later issues when computing likelihoods for new data. A coefficient from any test sample that happens fall into one of these bins will zero out all probability for that sample. This, in turn, can zero out all probability for any aggregated window of time in which that sample is included, regardless of what else happens within that window. When dealing with log-likelihoods, the log of an exact zero is often given as $-\infty$, which can cause NaNs to propagate through the results.

For these reasons, it is generally best to avoid hard zeros in a PDF estimate by applying some form of regularization. We do this by taking the amount of probability that one additional sample would represent, distributing it evenly amongst all the histogram bins, and re-normalizing the histogram so it integrates to one. When fewer non-zero samples are available for a particular atom, the effects of this redistribution of probability will be larger, matching our lower confidence in the original histogram because of the smaller number of samples used to estimate it. When a dictionary atom is never used over the training data, this redistribution of probability creates a uniform distribution over all coefficient values for that atom. The lack of sufficient samples to form a useful coefficient PDF estimate will be represented in the atom usage probabilities estimated in Section 6.6.1.

Situations where certain atoms are rarely or never used in the training coefficients $\mathbf{X}_\mathrm{t}$ are more likely to occur when the dictionary is trained on a broad set of data $\mathbf{Y}_\mathrm{d}$ and the training data $\mathbf{Y}_\mathrm{t}$ is selected narrowly, such as when trying to characterize a specific sound.

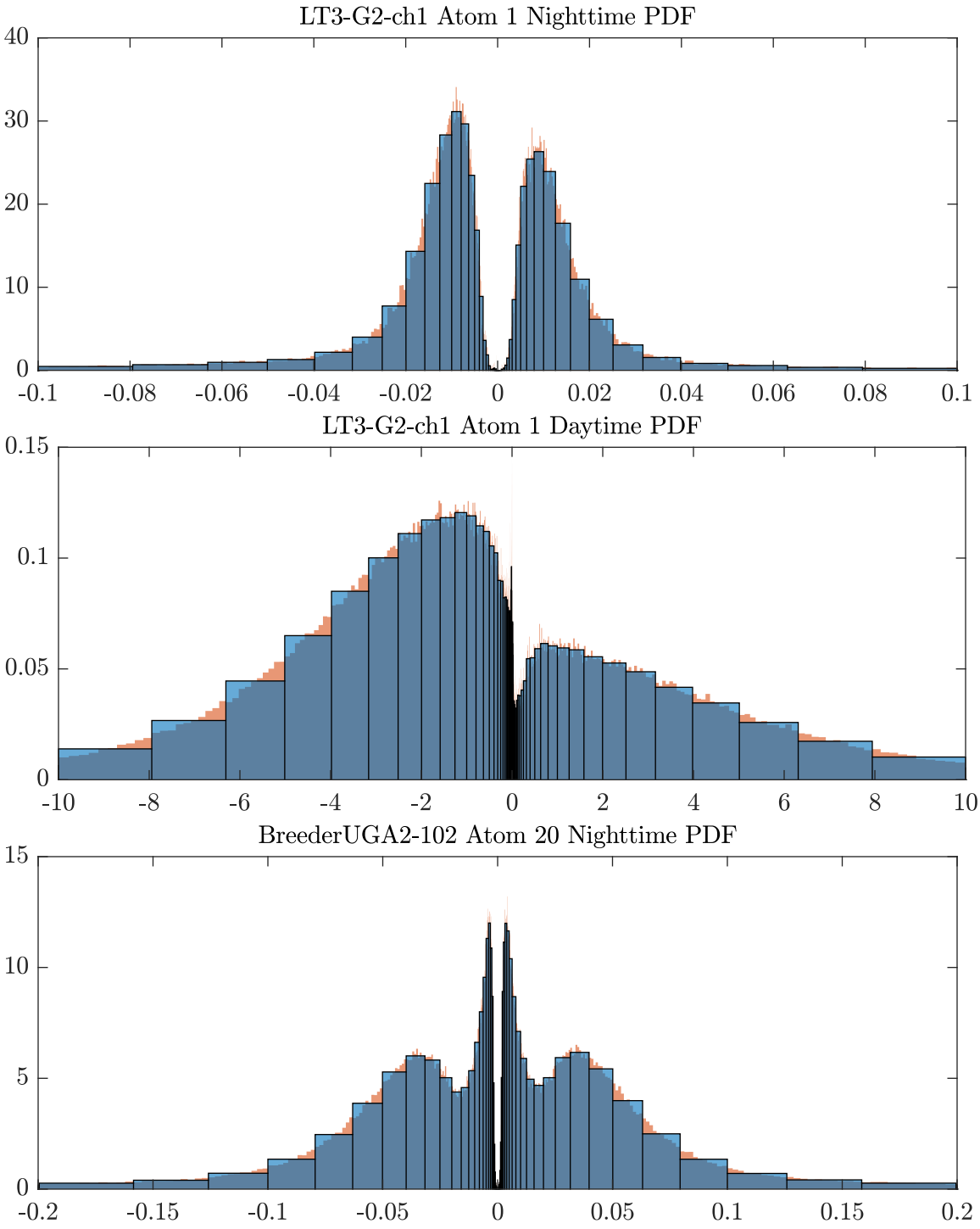

Figure 19: Sample coefficient PDFs from different datasets. The blue bars show the histogram that is used as the PDF estimate. The orange bars in the background show a histogram of the same data with ten times the bin resolution to give an idea of how well the blue histogram fits the underlying data. Note that the blue PDF estimates follow the shape of the distribution across all three plots, despite large differences in the typical coefficient magnitudes (as reflected by the large differences in x-axis scaling).

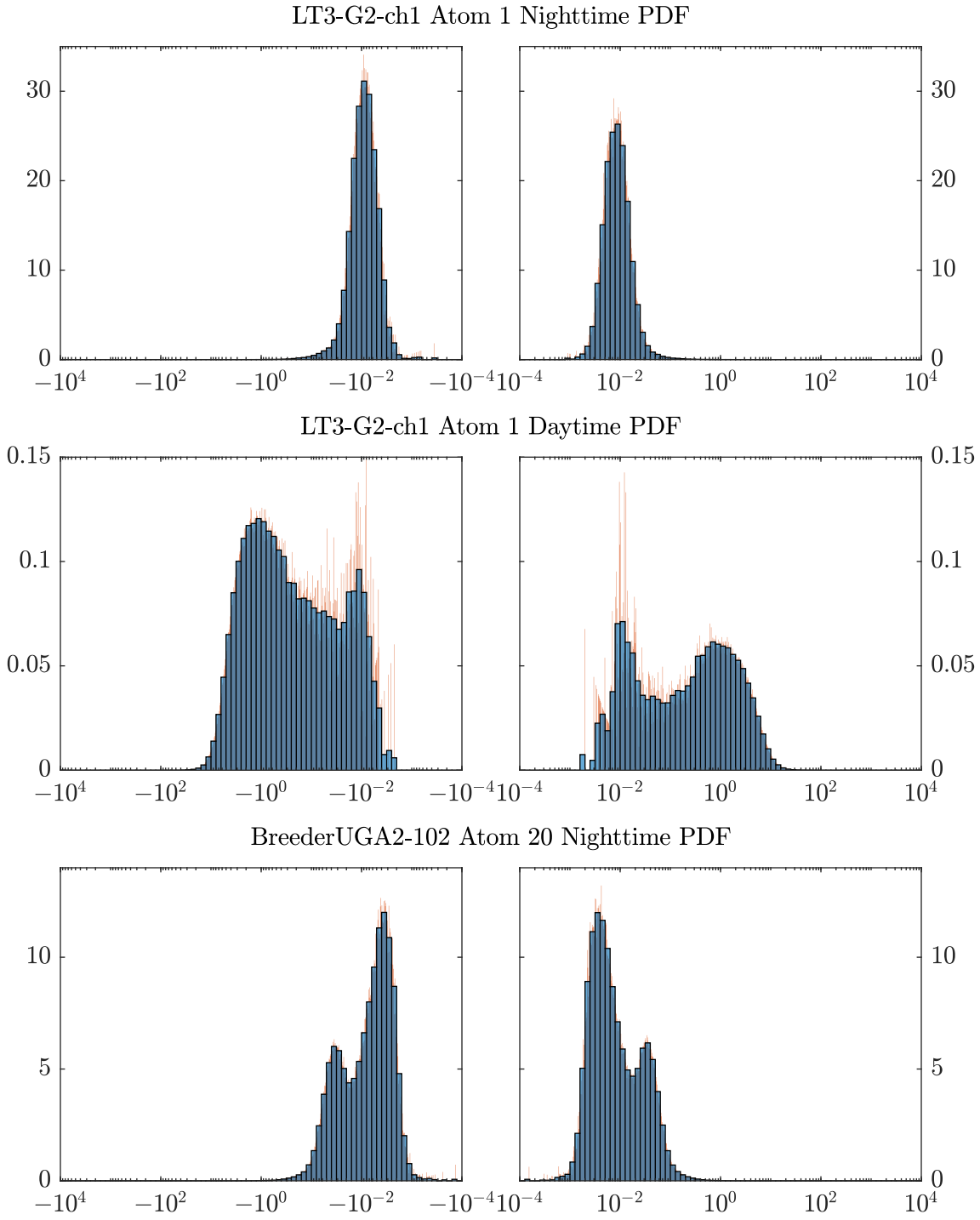

Figure 20: Sample coefficient PDFs with log scaling. The above plots show the same PDF estimates as are depicted in Figure 19, but with a log-scaled x-axis that makes all the bins clearly visible. Note that the plots are broken apart between -10^{-4} and 10^{-4} on the x-axis because a log-scaled axis cannot include zero. As before, the orange bars in the background represent a histogram of the same data with ten times the bin resolution. They are provided to show how well the estimate fits the underlying data.

6.6.3 Residue magnitude PDF

The residual error is computed by subtracting the reconstructed training data from the original training data, $\mathbf{E}_\mathrm{t} = \mathbf{Y}_\mathrm{t} - \mathbf{D}\mathbf{X}_\mathrm{t}$. We take the Euclidean magnitude of each column of $\mathbf{E}_\mathrm{t}$ as a distance measure between what the dictionary is able to represent and what is present in the given sample. Then we characterize these residual error magnitudes by estimating a PDF for them much like we did for the non-zero coefficient values.

There are two differences in estimating the PDFs for the residue magnitudes versus the coefficient values. First, every training sample is used in the residue magnitude PDF estimation since the residues are not sparse like the coefficient values. Second, only the bins representing non-negative numbers are used for the residue PDFs since magnitudes are always non-negative. Otherwise, the bin edge locations and estimation procedure are identical to what was done for the coefficient PDFs in Section 6.6.2.

Once we have obtained the histogram of the residue magnitude values for the training data from mode $\mathcal{M}$, we can use it directly as our estimate of the probability density $\mathrm{p}(r|\mathcal{M})$.

6.7 Log-likelihood estimation (step 4)

With the estimates of the atom probabilities $\mathrm{p}(a_i|\mathcal{M})$, the coefficient distributions $\mathrm{p}(c_{a_i}|\mathcal{M})$, and the residual error magnitude distribution $\mathrm{p}(r|\mathcal{M})$ obtained in the previous step, estimating the likelihood of new data $\mathbf{Y}$ is relatively straightforward. First, the data is sparse coded as $\mathbf{Y} \approx \mathbf{D}\mathbf{X}$. Then, the likelihoods of the atoms chosen, the likelihoods of their coefficient values, and the likelihood of the residue magnitude are looked up from the estimated distributions in the ELSA model. Finally, these likelihoods are multiplied together as given in Equation (9) in Section 6.6 to get an estimate of the likelihood of each sample. Under the assumption that all the samples are independent, the likelihood of windows of data can be computed by taking the product of all the samples in the window.

In practice, taking the product of a large number of probabilities can lead to numeric issues because the numbers get extremely small. It is common to use log-likelihoods instead

to make the results more numerically stable and easier to interpret. Denoting the log-likelihood of the data sample $\boldsymbol{y}_i$ as $L_{\boldsymbol{y}_i}$, taking the log of Equation (9) gives

$$L_{\boldsymbol{y}_i} = \log\left(\mathcal{L}(\mathcal{M}|\boldsymbol{y}_i)\right) = \log\left(\mathrm{p}(r|\mathcal{M})\right) + \sum_{j=1}^{s}\Big(\log\left(\mathrm{p}(a_j|\mathcal{M})\right) + \log\left(\mathrm{p}\big(c_{a_j}|\mathcal{M}\big)\right)\Big). \tag{12}$$

Computing the log-likelihood over a window of data $\mathbf{Y}$ with n samples similarly involves summing the log-likelihoods of each sample within that window:

$$L_{\mathbf{Y}} = \log\left(\mathcal{L}(\mathcal{M}|\mathbf{Y})\right) = \sum_{i=1}^{n} L_{\boldsymbol{y}_i}. \tag{13}$$

Note that the magnitude of the log-likelihood of a piece of data will be affected by the number of parameters present in that piece of data. For instance, with everything else equal, the log-likelihood of a sample of data with more non-zero coefficients (a larger sparsity constraint s) will tend to be smaller than the log-likelihood of one with a smaller s simply because more (usually negative) numbers are added together in its computation.

Similarly, the probabilities of the a_i and c_{a_i} parameters will outvote the probability of the single r parameter more when s is large. This effect is inherent in the independence assumption. However, it also makes sense because more information can be captured by the sparse coding parameters when more atoms can be used to encode a sample, leaving less information left over in the residue.

In order to facilitate comparisons of results across different settings of the s parameter, we prefer to calculate the mean log-likelihood across all the different random variables instead of the summed log-likelihood. This does not change the shape of the results, but rather scales them so that comparisons can be made even when the number of atoms used differs. Referring to Equation (12) and denoting the mean with an overbar, the mean log-likelihood can be calculated as

$$\bar{L}_{\boldsymbol{y}_i} = \frac{L_{\boldsymbol{y}_i}}{2s+1} = \frac{1}{2s+1}\left(\log\left(\mathrm{p}(r|\mathcal{M})\right) + \sum_{j=1}^{s}\Big(\log\left(\mathrm{p}(a_j|\mathcal{M})\right) + \log\left(\mathrm{p}\big(c_{a_j}|\mathcal{M}\big)\right)\Big)\right), \tag{14}$$

where $2s+1$ is the number of random variables present. $\bar{L}_{\boldsymbol{y}_i}$ can be thought of as the density of the log-likelihood per random variable for the ith data sample.

Similarly, we calculate the mean log-likelihood over the samples in a window of data to make the results invariant to the size of the window under consideration. Modifying

Equation (13) to get the mean over the samples gives

$$\bar{L}_{\mathbf{Y}} = \frac{1}{n} \sum_{i=1}^{n} \bar{L}_{\boldsymbol{y}_i}. \tag{15}$$

$\bar{L}_{\mathbf{Y}}$ can be thought of as the density of the log-likelihood per sample.

Since it can be more intuitive to think about how much data deviates from a trained model instead of thinking about its likelihood under that model, we often transform the log-likelihoods into a deviation measure simply by negating them. This causes normal behavior to show up as smaller numbers, while abnormal behavior will show up as larger numbers. Most of the plots in this work use this deviation measure instead of the log-likelihood values.

6.8 Decision making (step 5)

How decisions are made based on the results from the ELSA method will be highly dependent on the application and on the approach taken. We provide some ideas for common situations here, but there are undoubtedly other ways the ELSA method could be applied.

If ELSA is being used primarily for anomaly detection, the simplest approach would be to set a threshold on the deviation from a model trained on the normal behavior of the system. Statistics for the deviation values could also be calculated over the training data and used to set the threshold, or used as a baseline against which to compare statistics calculated from the new data. Instead of setting a hard threshold, deviation values could also be compared against the trend over the surrounding window of time. Algorithms to detect peaks or sudden spikes also might be appropriate in some circumstances.

For classification tasks, the likelihoods of multiple different models could be compared against each other to determine the most likely class. If none of the models seem likely, that could indicate that an anomaly has occured or that the system is operating under a new mode for which a model has not been learned. If multiple models seem likely, that could be an indication of overlapping classes or modes. The likelihood values from different models could also be used as features passed into other classification or learning algorithms.

For all of the above methods, the deviation values can be aggregated over different windows of time in order to target deviations of a certain length or to reduce the effects

of noise in the results. Or, classification can be done based on statistics calculated on the deviation values over a window of time.

In some cases, the user may be more interested in seeing plots of the deviation or likelihood values instead of relying on an algorithm to make a classification. In our work with poultry data, we have found that it is useful to plot each day of data as a row in a heatmap, where the days are indexed down the y-axis and the time of day is indexed along the x-axis. For poultry data, we call these plots growout assessment maps (GAMs) and find that they can provide a nice visual overview of what happened while raising a flock. This type of plot also makes it easier to see daily trends in the data since things that happen around the same time of day line up vertically. When we have shown these plots to farm managers, they have often immediately started pointing out where different things that happen during the day show up in the plot. Example GAMs are given in Section 6.10.

If things drift over time, the dictionary and model may occasionally need to be retrained or updated. To encourage continuity in what each atom represents, the current dictionary could be used to initialize the dictionary training.

6.9 K-SVD versus PCA for artificial anomaly detection

The ELSA method presented in this chapter relies on sparse coding of the underlying feature space. A question that may arise in evaluating the effectiveness of this method is whether or not the sparse coding step provides any value over other methods of reducing the dimensionality of the data. In this section, we compare the ELSA method against a modified algorithm that uses principal component analysis (PCA) for dimensionality reduction instead of sparse coding. For both algorithms, anomalies are detected by comparing the algorithm's estimated log-likelihood of each sample against a threshold.

6.9.1 Modified algorithm using PCA

Changing the ELSA method to use PCA instead of K-SVD required three modifications to the steps outlined in Section 6.3:

1. PCA is run on the data selected for dictionary learning instead of K-SVD. The mean value of each feature over the training set is stored since the means are subtracted

off as part of the PCA operation (unlike for K-SVD). The extracted principal component basis set is then reduced by selecting only the basis vectors that span the most variance in the selected data. The number of basis vectors retained is equal to the sparsity constraint used for K-SVD, giving both methods the same number of non-zero coefficients to base decisions on. For the results presented here, a sparsity constraint of 5 was used.

2. The sparse coding (OMP) step applied to the test data and to the data used to learn probability distributions is replaced with a projection of that data (minus the stored means) onto the reduced principal component basis set.

3. The stored mean values for the features are added back in to the reconstructed signal before subtracting it from the original features to calculate the residual error.

The remaining steps in the modified algorithm (estimating probability distributions and calculating log-likelihoods) are identical to those for the ELSA method. However, it should be noted that the dense representation given by PCA turns all of the atom usage probabilities described in Section 6.6.1 into constant ones for the modified algorithm. Thus, those probabilities will neither contribute to nor detract from the discriminative power of the modified algorithm.

6.9.2 Test data

Due to the nature of this work, it is difficult to get definitive quantitative results on the performance of our algorithms—particularly for tasks such as anomaly detection. This is primarily because the vast majority of our data is unlabeled, but also in part because the question of exactly what should and should not constitute an anomaly is highly subjective and task-dependent.

To reduce subjectivity and avoid the need to label large amounts of data, we artificially inserted anomalies into sets of data extracted from the commercial (COM) dataset to test how well the algorithms can detect them. Each set of data was generated using 60 one-minute-long recordings randomly selected from either daytime or nighttime hours. Once the

files were selected, we made 11 copies of them with anomalies added at SNRs ranging from -20 dB to 20 dB in steps of 4 dB. The anomaly locations and durations were randomized for each of the 11 versions of the data.

The above process was repeated to generate 11 sets of data each for two different anomaly types during both daytime and nighttime hours. The first type of anomaly consisted of snippets randomly extracted from a recording of an empty augur running in the facility at the University of Arkansas. Although augur sounds are not uncommon and the augurs used in Arkansas are likely similar to those used in the COM data, the fact that it was empty changes the sound and makes it anomalous. The second type of anomaly consisted of snippets extracted from a recording of constant, excited chirping caused by a worker's presence in the COM dataset. Although a sound completely foreign to the environment could have been used, these choices more accurately reflect anomalous sounds from the equipment and from the birds that realistically might occur.

We also tested a third anomaly type that consisted of excited chirping (caused by a worker's presence) from when the birds were very young. The results from this test were consistent with those for the other two anomaly types, and have been omitted for brevity.

All of the inserted anomalies were faded in and out over 0.01 seconds using the corresponding halves of a Hanning window. They also randomly varied in duration from 0.5–3.0 seconds. Between two and eight anomalies were inserted into each minute-long file.

The gain applied to the anomaly clips to get the desired SNR was dependent on the relative average power levels of the signal immediately surrounding the regions of interest. We calculated the average power of both signals over a window spanning from two seconds before to two seconds after the segments where the anomaly would be extracted from and inserted into. Estimating the power locally prevents other sounds in the recordings that are not temporally close to the location of the insertion from affecting the insertion gain.

Adding anomaly waveforms into the existing recordings can cause clipping if the magnitude of the summed signal exceeds 1.0. The gain of the summed signal could be adjusted to avoid clipping, but such an adjustment would to some degree invalidate the estimated probability distributions for that data. Thus, we allowed clipping to occur when inserting

anomalies. However, the COM data is recorded at relatively low levels, so clipping only occurred when the anomalies were inserted at the highest couple SNR settings.

When selecting files from the COM dataset, no effort was made to avoid anomalous sounds that might already be present in the COM data. If such an anomalous sound gets flagged by the algorithm and does not overlap with an artificially inserted anomaly, it is counted as a false positive. Furthermore, the boundaries of the analysis windows and inserted anomalies do not line up. We set the ground truth values based on majority overlap, but misclassifications due to partial overlaps still have some potential to slightly degrade the performance numbers. Thus, the results reported here should generally be pessimistic.

6.9.3 Model learning

We evaluated the performance of nighttime and daytime models trained using both the ELSA method and the modified PCA-based method. Only one K-SVD dictionary was learned, and this dictionary was shared across both the nighttime and daytime ELSA models. Similarly, the nighttime and daytime PCA-based models shared the same reduced PCA basis set.

The K-SVD dictionary and the PCA basis set were each learned using 48 hours of data sampled randomly from the middle 27 days of the COM dataset. Both daytime and nighttime recordings were included in this sampling. The features used were the delta and delta-delta features described in Section 3.2.5.

Using the above dictionary and basis set, the daytime and nighttime models were learned by estimating the probability distributions described in Section 6.6 over 48 hours of data respectively sampled from the daytime and nighttime hours of the dataset. Once these models had been obtained, they were used to compute the mean log-likelihood for each sample as specified in Equation (14) from Section 6.7. These values were then compared to a threshold to classify each sample as anomalous or not.

6.9.4 Performance comparison

To evaluate the ELSA method against the modified PCA-based method, we swept a detection threshold across the full range of values to generate a receiver operating characteristic (ROC) curve for each test configuration. We then computed the area under the ROC curve

Table 6: AUC performance comparison of the ELSA method using K-SVD based features versus PCA based features for an anomaly detection task. These results are also plotted in Figure 21.

SNR (dB)	Augur Anomaly				Chirping Anomaly			
	Nighttime		Daytime		Nighttime		Daytime	
	PCA	K-SVD	PCA	K-SVD	PCA	K-SVD	PCA	K-SVD
-20	0.533	0.528	0.488	0.491	0.497	0.505	0.485	0.493
-16	0.575	0.563	0.465	0.467	0.555	0.560	0.507	0.520
-12	0.641	0.625	0.487	0.484	0.622	0.624	0.516	0.531
-8	0.709	0.684	0.510	0.518	0.687	0.686	0.545	0.573
-4	0.793	0.779	0.547	0.566	0.759	0.753	0.591	0.632
0	0.867	0.866	0.598	0.636	0.859	0.855	0.647	0.694
4	0.919	0.930	0.681	0.732	0.930	0.931	0.757	0.798
8	0.946	0.962	0.780	0.840	0.961	0.966	0.809	0.847
12	0.961	0.975	0.853	0.906	0.978	0.983	0.893	0.920
16	0.969	0.978	0.919	0.953	0.986	0.988	0.930	0.951
20	0.974	0.981	0.948	0.969	0.988	0.989	0.965	0.975

(AUC), which is a commonly used metric for to evaluating the discriminative power of a system. An AUC of 0.5 is equivalent to a detector that guesses randomly, while an AUC of 1.0 represents a detector that can perfectly classify the data.

The AUC metrics are listed for each test configuration in Table 6 and are plotted in Figure 21. Individual ROC curves for several of the configurations are plotted in Figures 22–25.

The clearest trend from Figures 21–25 is that K-SVD consistently outperforms PCA on the daytime datasets, whereas their performances on the nighttime data are roughly similar. The daytime data poses a much more difficult problem than the nighttime data because there is much more variance (and often energy) in the sounds present. The chickens make noise by vocalizing, feeding, drinking, flapping their wings, running around, etc. Noises from human workers checking on the birds and maintaining equipment are also far more likely to occur during the day. At night, there may be occasional activity from the chickens, but the soundscape tends to be dominated by fan and equipment noise.

The results indicate that the sparse coding model is a better fit for the daytime data than the PCA model. Sparse coding's ability to select a small, customized basis set for

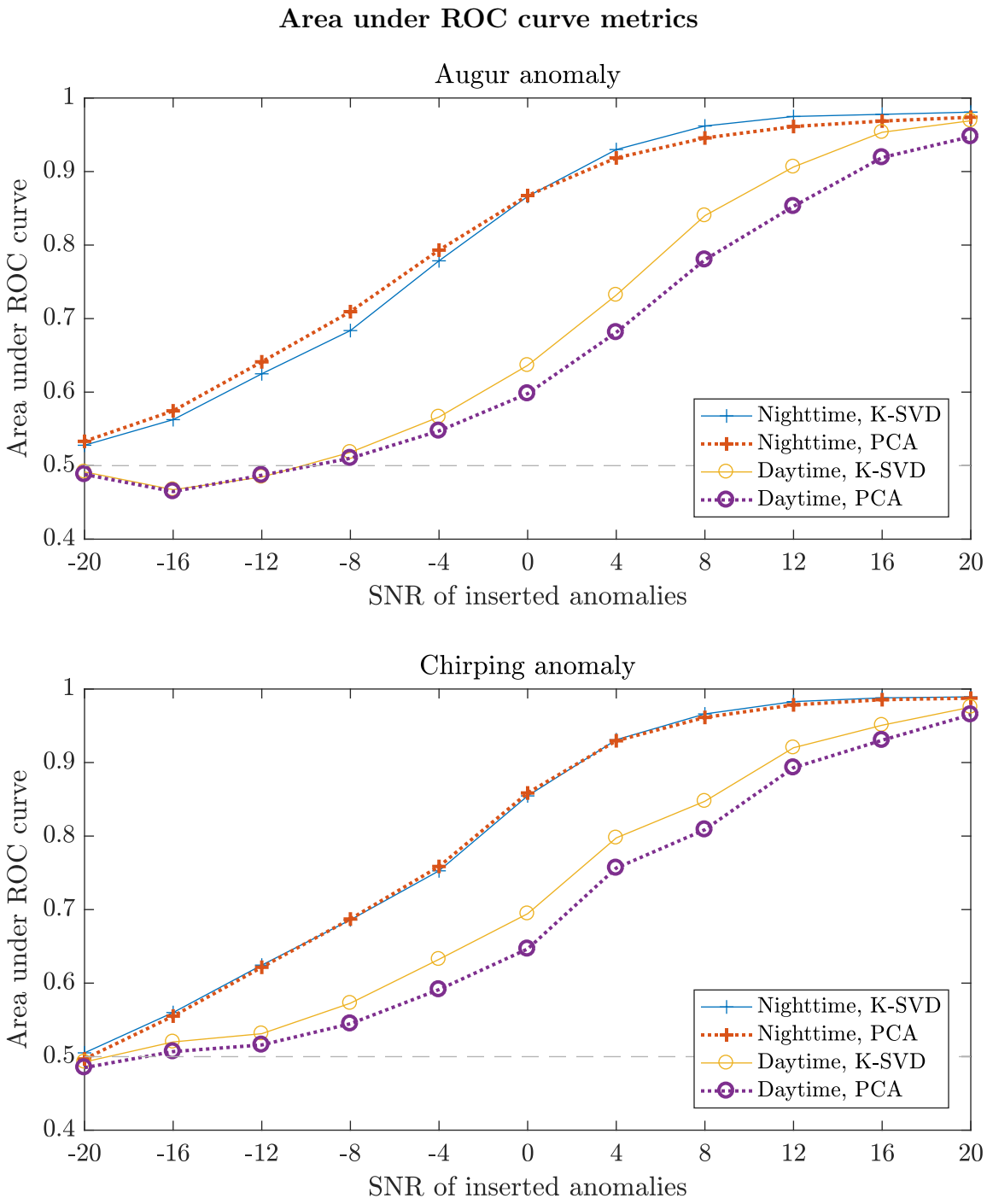

Figure 21: AUC performance comparison of the ELSA method using K-SVD-based features versus PCA-based features for an anomaly detection task. The nighttime and daytime results for the augur anomaly are given in the top plot, while those for the chirping anomaly are in the bottom plot. The horizontal reference lines at 0.5 mark the expected AUC metric for a predictor with chance performance. The K-SVD-based model consistently outperforms the PCA-based model on the more difficult daytime dataset, except at the lowest SNRs where the performance is no better than chance. On the easier nighttime dataset, both models score higher, but the differences between their scores are much smaller.

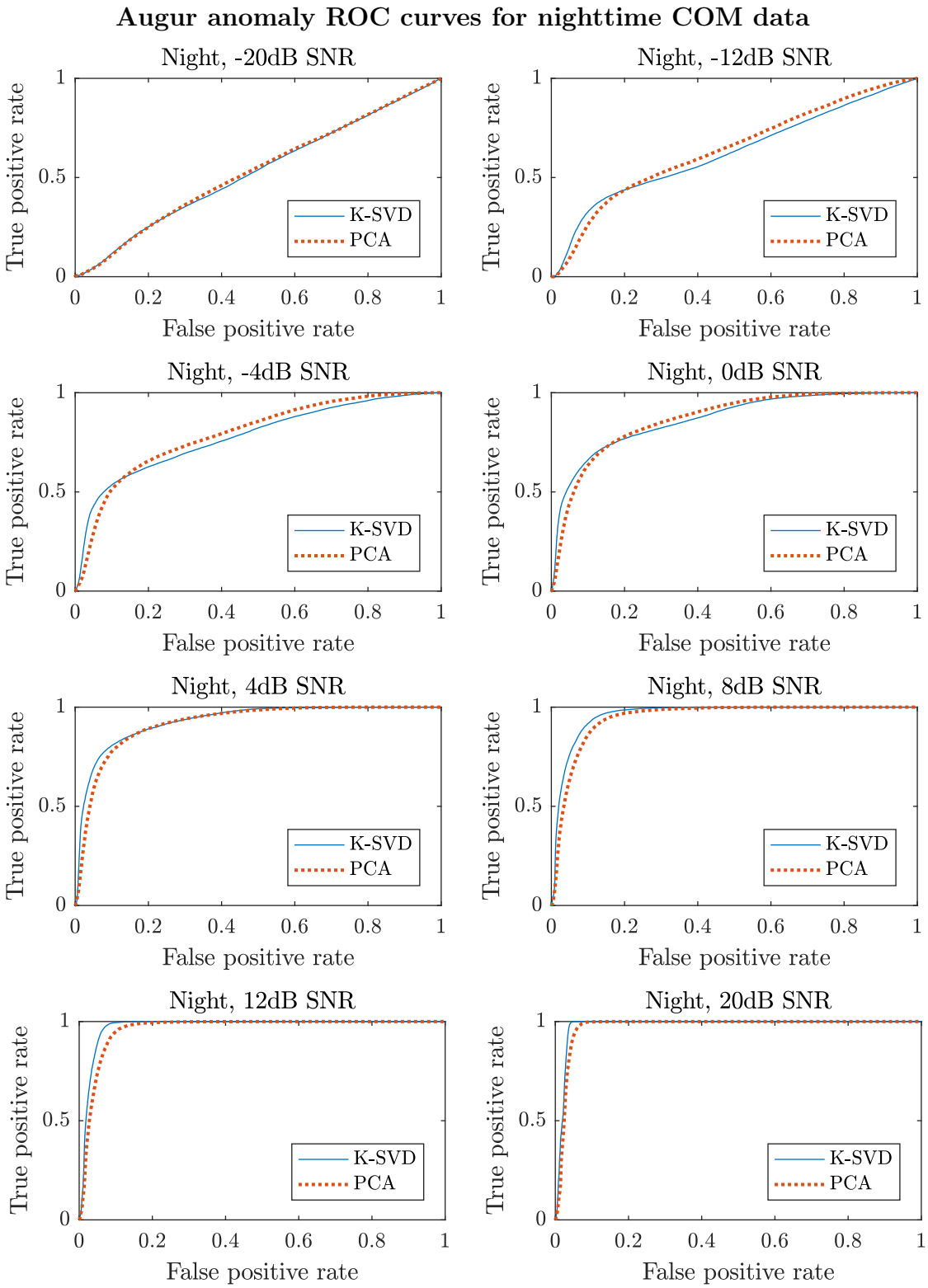

Figure 22: ROC plots for nighttime augur anomaly detection using COM data. Note that for almost all SNR levels, K-SVD starts with a slight edge over PCA for the less sensitive thresholds in the lower left corner of the plots. However, for low SNR levels, PCA overtakes K-SVD toward the upper right portion of the curve (with more sensitive thresholds).

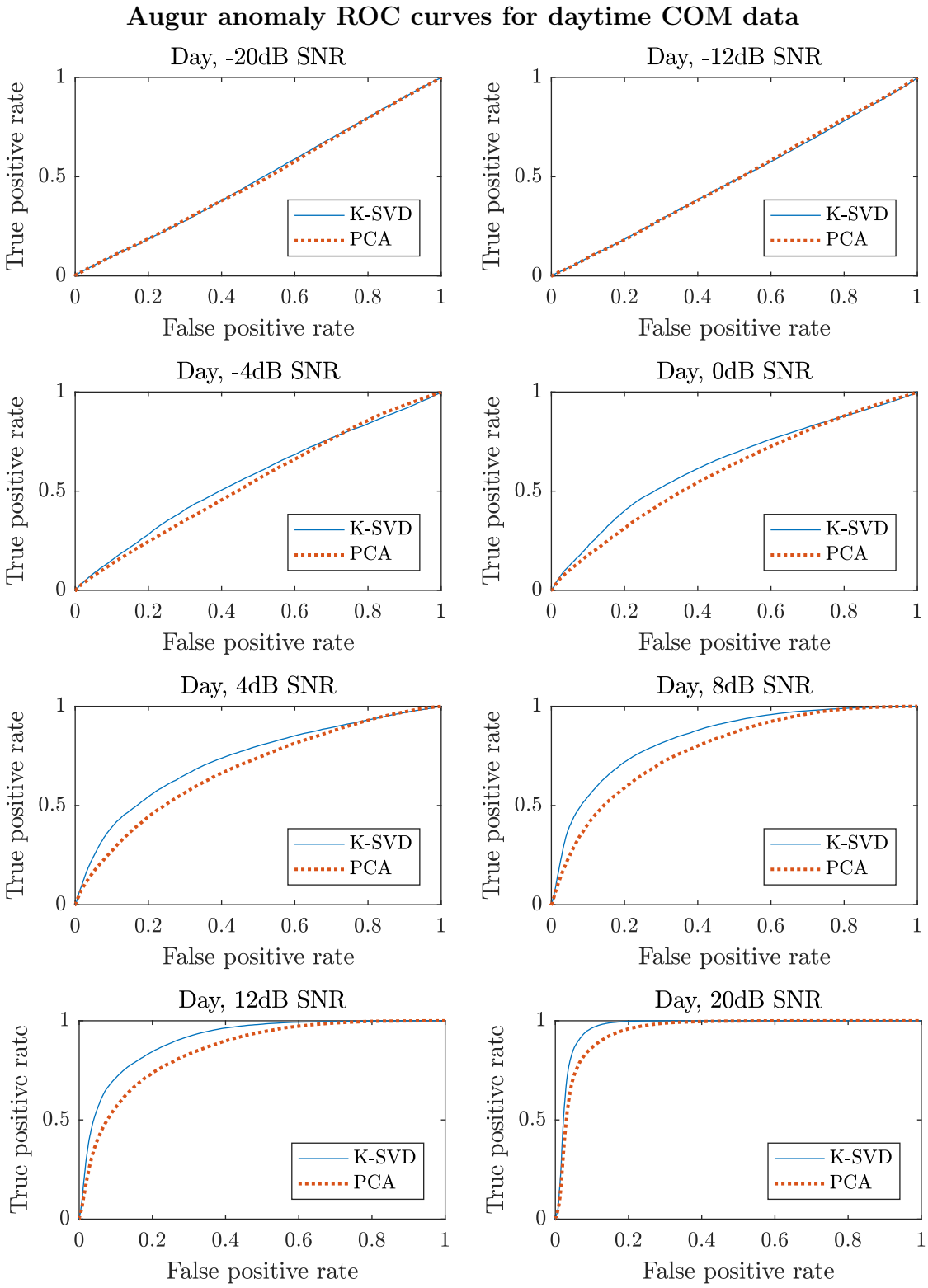

Figure 23: ROC plots for daytime augur anomaly detection using COM data. K-SVD consistently outperforms PCA on the daytime data except at the lowest SNR levels where neither method performs much better than chance.

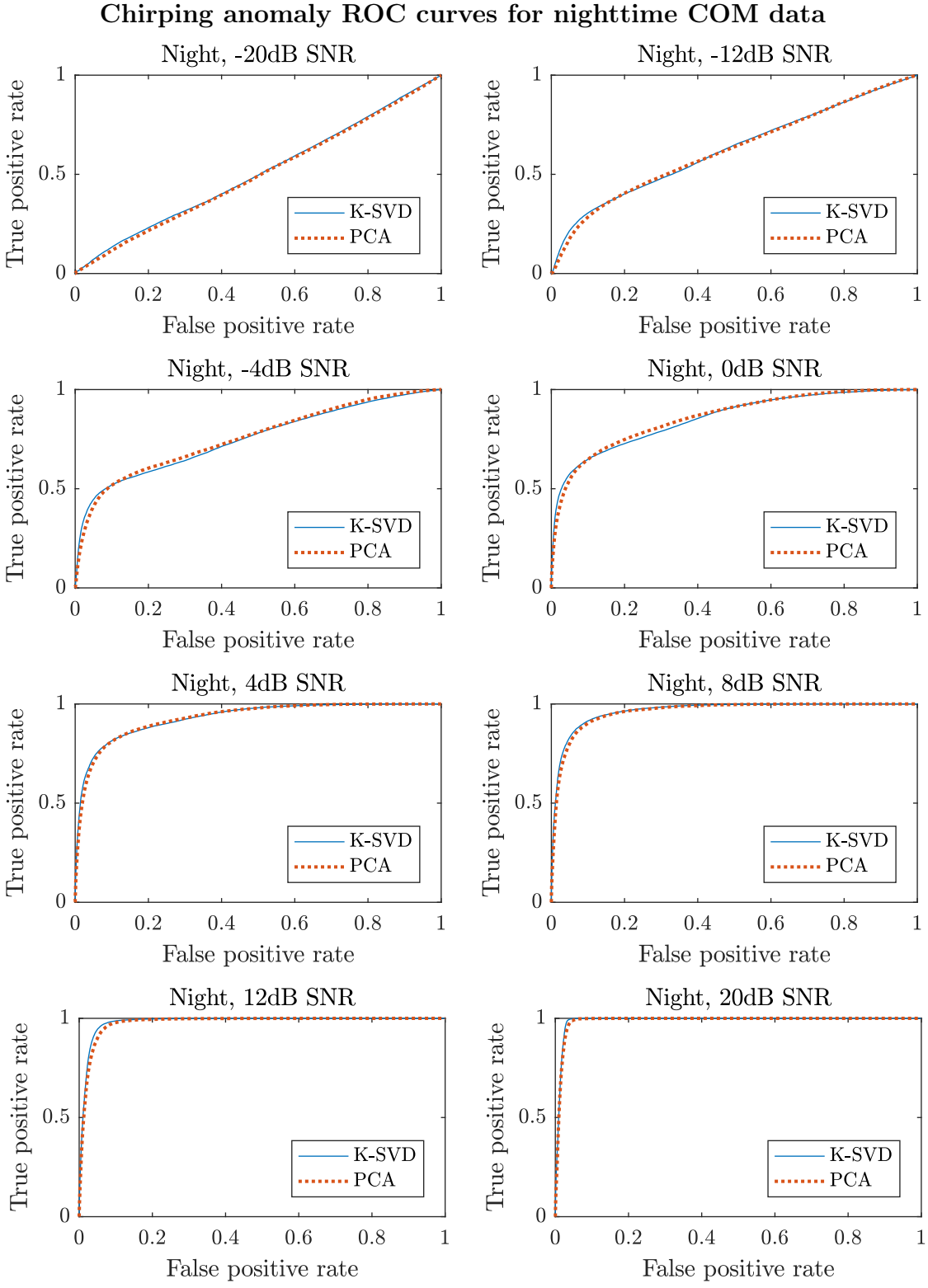

Figure 24: ROC plots for nighttime chirping anomaly detection using COM data. K-SVD and PCA yield similar results across the board for this data.

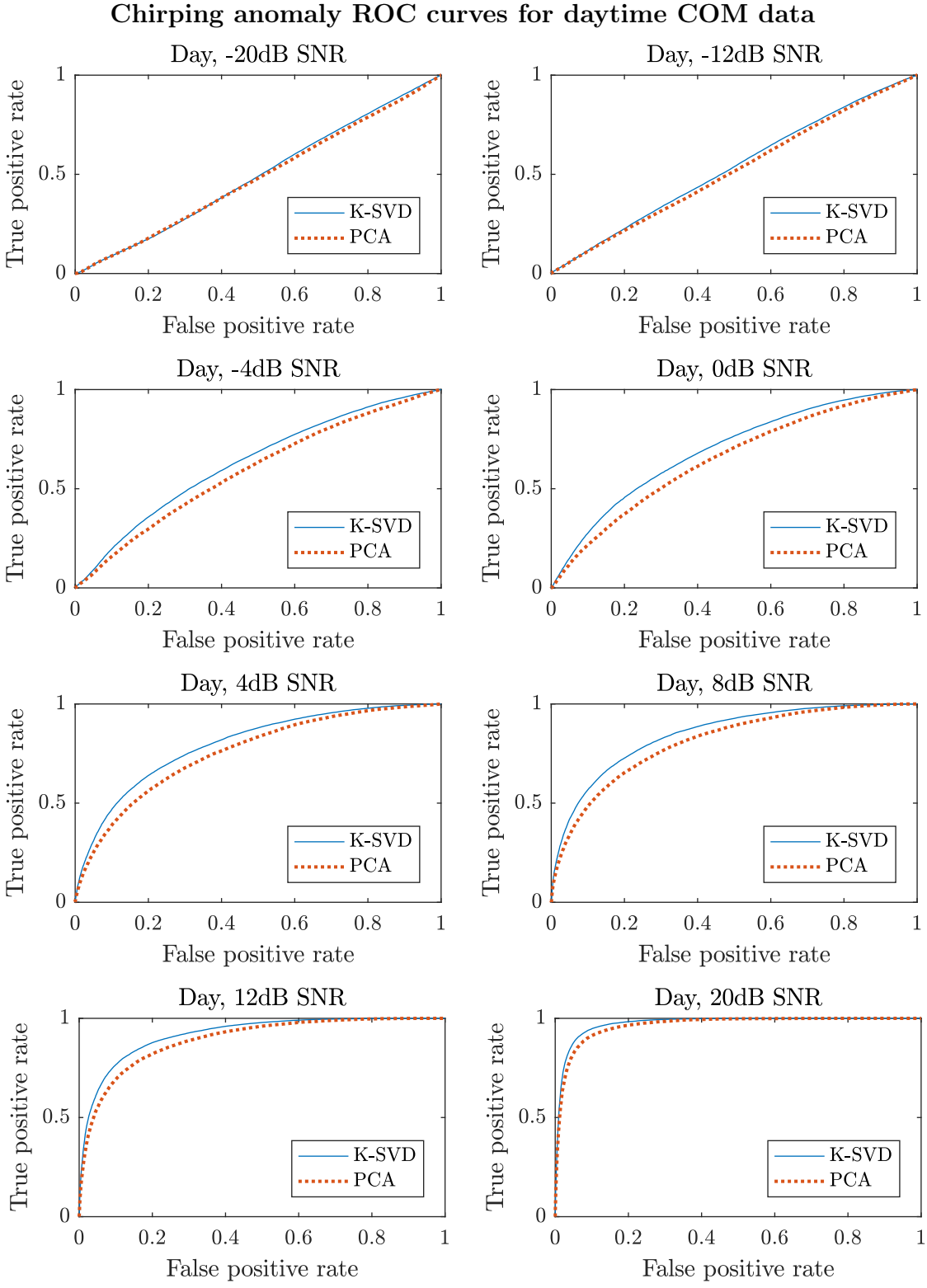

Figure 25: ROC plots for daytime chirping anomaly detection using COM data. K-SVD once again consistently beats PCA on the daytime data.

each sample allows it to represent a larger variety of sounds, including those that are only sporadically present. The PCA representation is limited to using the same few basis vectors for every sample, and will tend to focus on representing the sound structures that are most constantly and consistently present across all the samples in the training data. It is thus not as well equipped to handle a wide variety of normal sounds that come and go sporadically over time, like many of the sounds that are present during the daytime data. At night, the sporadic sound sources (primarily chickens and humans) are mostly dormant, and the data is dominated by the more constant equipment sounds. Since the PCA representation is good at representing that type of data, its nighttime performance is more on par with the K-SVD model.

It is also interesting to note that at SNR settings below 0 dB, the PCA model manages to outperform the K-SVD model in some cases. This may seem odd because the sparse representation can use just as many basis vectors as the PCA representation, but has the extra flexibility of choosing which ones to use. However, that extra representational power may be more of a disadvantage in situations where the noise is very regular and the anomalies are very quiet.

Unlike PCA, K-SVD is not optimal. The complexity of the sparse representations makes it computationally infeasible to find the optimal dictionary or coefficient matrix. So, we use sub-optimal algorithms like K-SVD that hopefully work well enough. In situations where the data is well matched to PCA's assumptions, its optimality may give it an edge over K-SVD in teasing out more subtle anomalies.

There is also some potential that the ELSA model may introduce a small amount of noise in the representation when dealing with data that is simpler than what it is set up to handle. If the information in a given sample is well represented by the first couple basis vectors selected during sparse coding, then any remaining basis vectors selected until the sparsity constraint is met may be chosen somewhat randomly as OMP matches them against the residual noise. This could introduce some noise into ELSA's atom usage and coefficient probabilities described in Sections 6.6.1 and 6.6.2, which might partially mask the subtle influence of quiet anomalies.

6.10 Results

The ELSA method has proven useful in locating anomalies or events across a wide variety of poultry datasets. Figures 26–37 show growout assessment map (GAM) plots from several of these datasets selected to highlight how different conditions and events affect the results of the ELSA method. These include lights turning on and off, disturbances in the chicken house, sickness, excessive heat, abnormal equipment noise, microphone gain changes, and other events. Although we have other datasets not represented here, the results for them are generally similar. After obtaining these results, we examined the data and logs corresponding to hot spots or other interesting features in the plots and added annotations indicating what we found. Periods of time when data is missing are represented with values of zero in the GAMs.

Unless stated otherwise, all the plots are of the minute-by-minute mean log-likelihoods (negated so that brighter spots represent more anomalous segments), utilized a dictionary containing 50 atoms with a sparsity constraint of 5, and used delta and delta-delta features. Similarly, the dictionaries and ELSA models were trained on up to 48 hours of data randomly selected from the specified periods of time. Most plots are of ELSA models trained on nighttime data because we have found them to provide good contrast in visualizing the levels of chicken activity compared to times when they are resting. To simplify navigation of the document, comments about the results are included in each of the figure captions.

6.11 Observations

In our experience using the ELSA method with poultry data, the coefficient likelihoods tend to dominate the shape of the final result. The atom probabilities usually have a much smaller variance, and thus have less of an impact on the shape of the results. When the model is trained on a narrow set of sounds, the variance in the atom probabilities is greater, but still tends to be smaller than the variance of the coefficient likelihoods. The residue likelihood numbers typically follow the same general shape as the coefficient likelihoods.

Because of this, we can often glean the same information from looking at plots of just the coefficient log-likelihoods instead of the full log-likelihood estimates. This may be in part

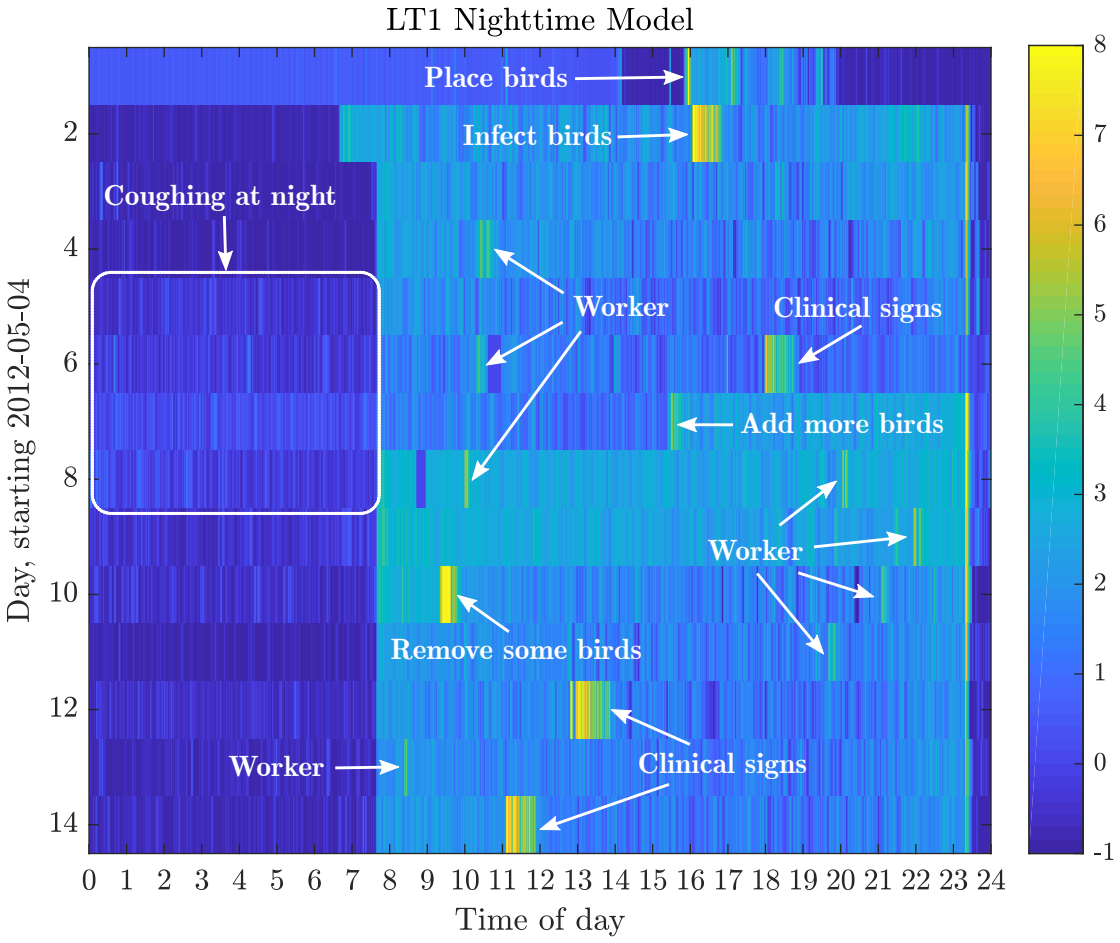

Figure 26: GAM for the LT1 (disease) data using a nighttime ELSA model. Each row in the heatmap represents a day of the experiment, with the time of day increasing across the x-axis. The color or brightness of the heatmap represents the negative mean log-likelihood calculated by the ELSA method over minute-long windows. Thus, brighter spots represent more anomalous events.

The dictionary was trained on data randomly selected from all of day 3, and the ELSA model was trained using nighttime data from day 3. We chose this time because it was prior to when symptoms of the disease would show up, but after the birds had some time to acclimate to the environment.

The nighttime hours when the lights were off are clearly visible as the darker block in the plot extending from after hour 23 until just before hour 8 of each day (other than the first two). The events where workers entered the room and handled the birds are clearly visible as the brightest spots. The birds were infected with LT on day 2, and were most strongly exhibiting symptoms of the disease on days 5 through 8. The brighter band during the nighttime hours for these days corresponds to significant coughing activity caused by the disease. Many additional chickens were present in the room from the afternoon of day 7 through the morning of day 10 to test transmission of the disease. The increased activity caused by the additional chickens is visible as a brighter band in the plot during that time.

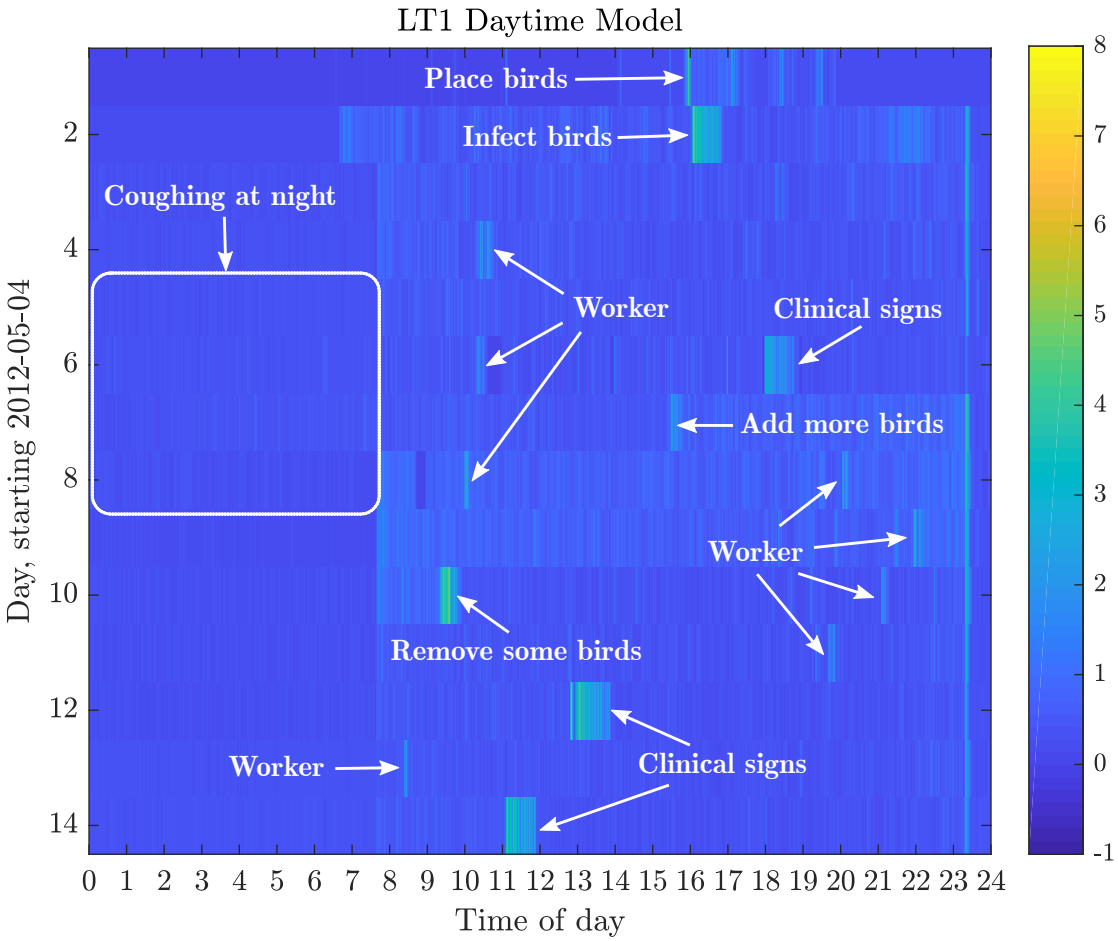

Figure 27: GAM for the LT1 data using a daytime ELSA model. The same dictionary was used here as for Figure 26, but the ELSA model was trained over the daytime hours instead of the nighttime hours of day 3.

Many of the same trends and events are visible here as compared to the nighttime model, though with less contrast since this model expects higher, daytime levels of activity. Compared to the nighttime model in Figure 26, the nighttime hours here are more anomalous and the daytime hours are less anomalous. Thus, the nighttime and daytime hours could be classified simply based on which of the two models is more likely.

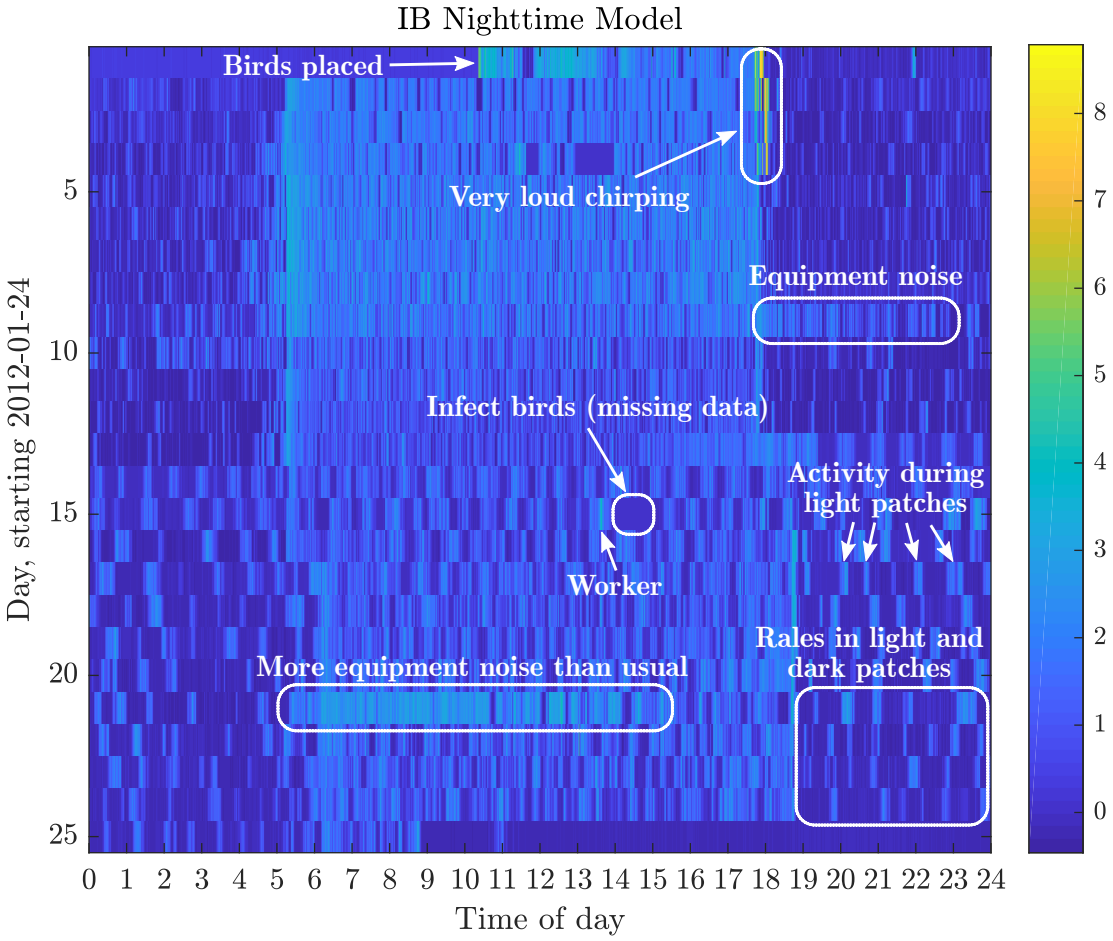

Figure 28: GAM for the IB (disease) data using a nighttime ELSA model. The dictionary and ELSA model were both trained on nighttime data from before the chickens were infected on day 15.

The chickens in the IB dataset exhibited the strongest symptoms of the disease over the last 5 days before the experiment was ended. In this case, the plot does not appear to show any indication that something was different over that time period. This is likely because the rales produced by the sick chickens are very subtle, while the background equipment noises are louder here than in any of our other datasets. The small, enclosed metal isolator boxes may have served to amplify equipment sounds, many of which were reminiscent of tennis shoes banging around in a tumble dryer.

Although this plot fails to differentiate between the healthy and sick periods, the brightness of the heatmap still correlates well with the amount of chicken activity and banging noises from the equipment. The rales are probably too quiet to significantly affect the sparse coding of the data, and thus its likelihood calculated through the ELSA method. In this case, the OLAF method covered in Chapter 5 produces better results.

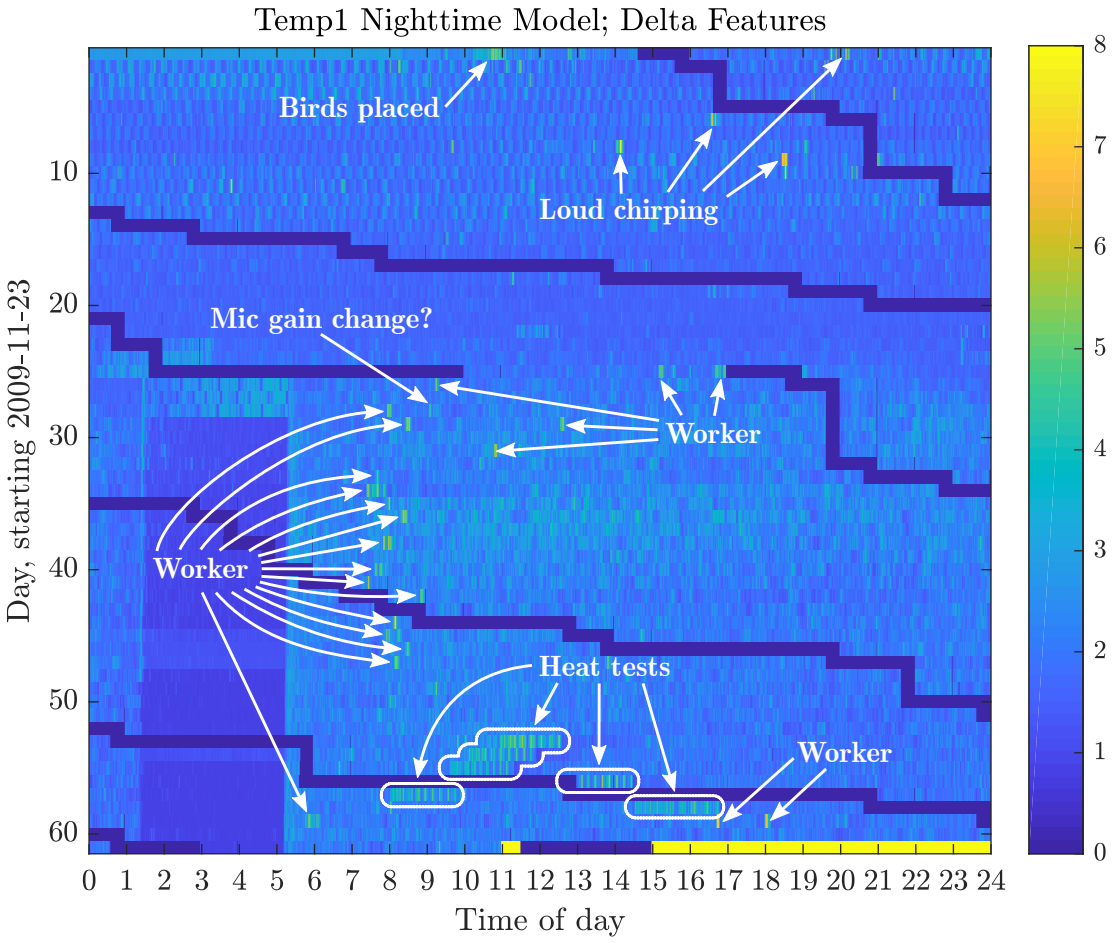

Figure 29: GAM for the Temp1 (heat test) data. The dictionary was trained on random data selected from the week before the heat tests (days 46–52), and the ELSA model was trained over the nighttime hours (2AM–5AM) from the same period of time. The dark patches snaking across the plot represent periods of missing data.

Six heat tests were performed on days 53–58 where the temperature was raised about 15–20 degrees Fahrenheit to make the birds uncomfortable. These tests each lasted about two hours, and are clearly visible in the plot as the blocks of brighter color bands on those days. All of the heat tests are immediately followed by a slightly dimmer spot in the plot, possibly due to the birds recovering from the heat before resuming normal activity levels.

Other bright spots in the plot generally correspond to increased chicken activity due to the presence of a human worker, or to loud chirping when the birds were young. The microphone gain was likely adjusted on day 28 (this is more visible in Figure 30), making the ELSA model less valid before then and causing the nighttime hours to look abnormal on days 26-28.

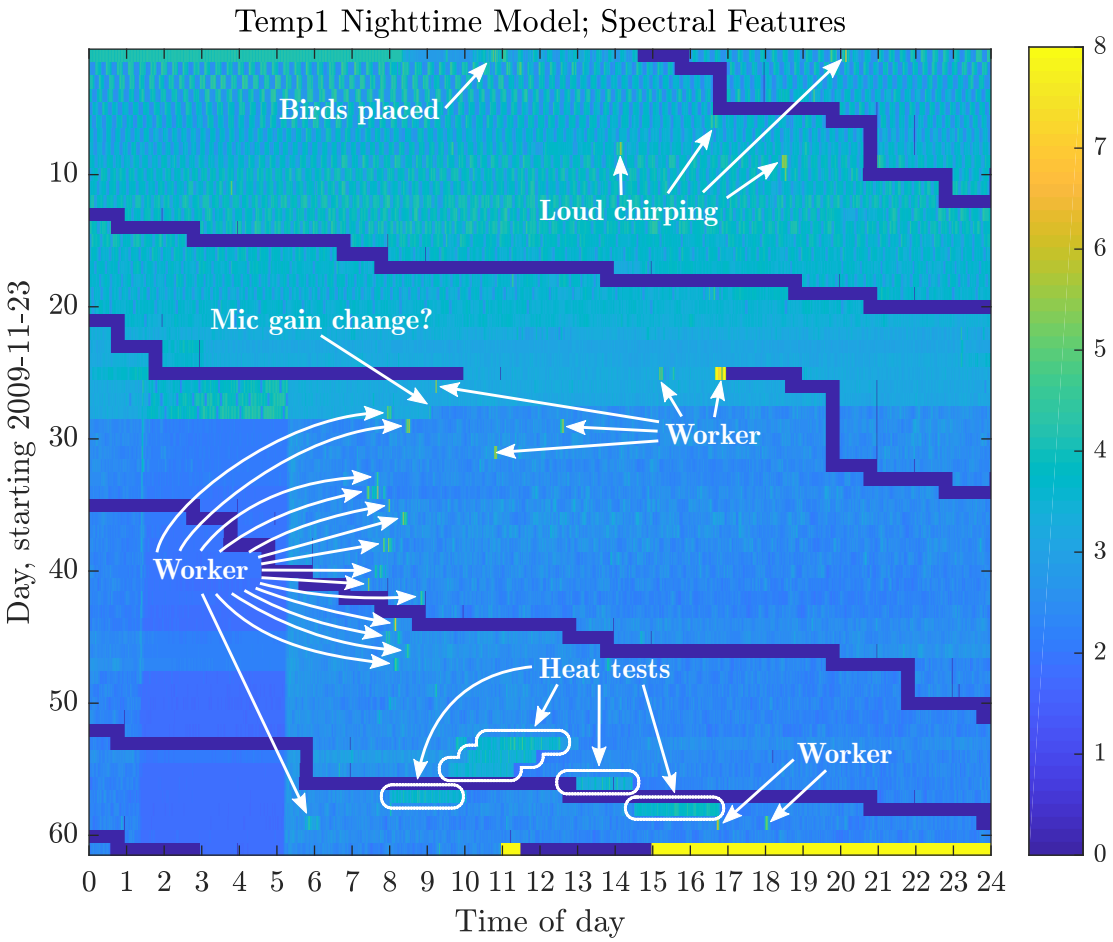

Figure 30: GAM for the Temp1 data using spectral features instead of delta features. The same time periods were used to train the dictionary and ELSA models as for Figure 29.

The heat tests show up in this plot as more constant hot spots instead of having alternating bright and dark spots as they did for the delta features in Figure 29. The general shift from brighter to darker on day 28 around 9AM is likely due to a microphone gain adjustment. Many of the same events are visible in this plot as in the previous. However, we generally prefer to use delta and delta-delta features because they are good at rejecting the more continuous noises that typically come from fans and equipment, while still responding to noises produced by the chickens.

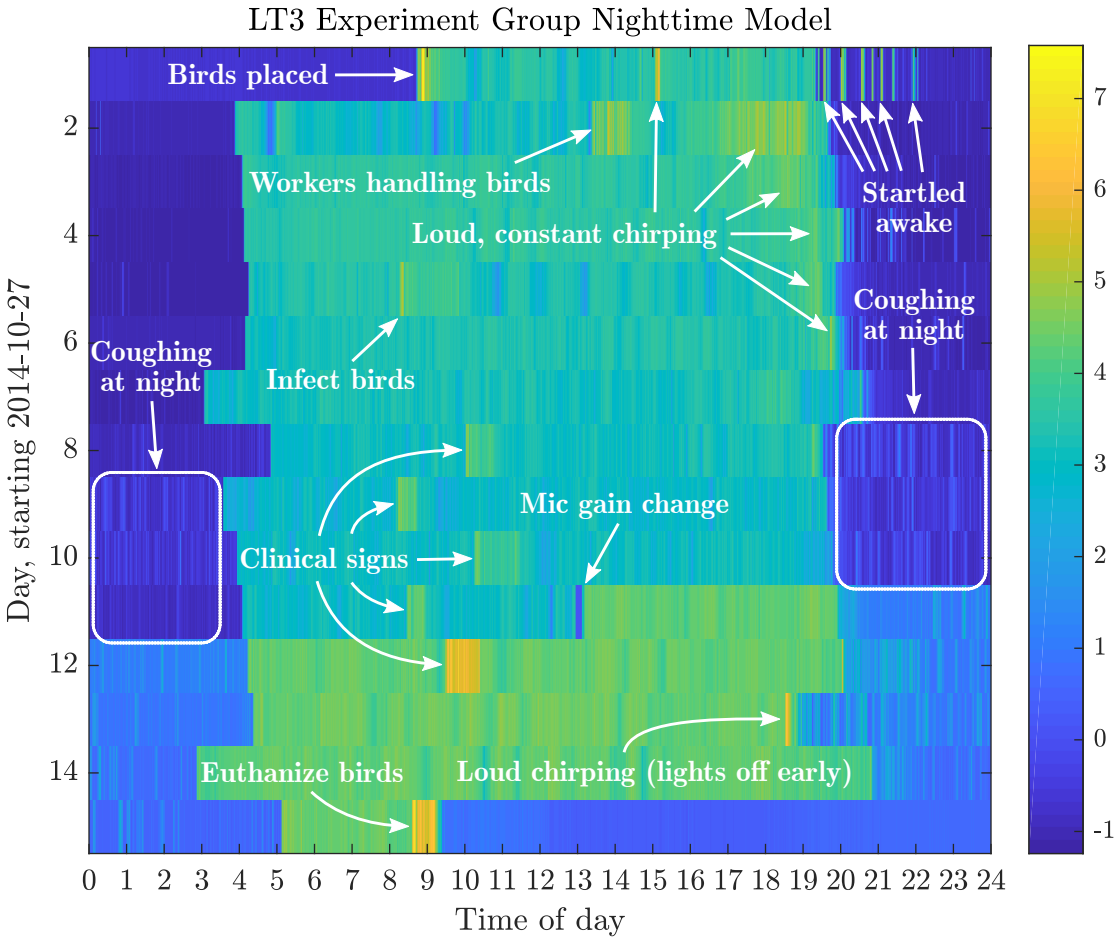

Figure 31: GAM for LT3 (disease) experiment group data showing an erratic lighting schedule and a mic gain change. The chickens in this room received a smaller dose of the vaccine under trial, but still got sick. The dictionary was trained on all the data prior to when the birds were infected on day 5, and the ELSA model was trained using nighttime data from the same period.

Prior to generating our first GAM for this data, we were unaware that the lighting schedule was erratic in this room. We have since verified from camera data that the step changes between the lighter and darker regions of the plot correspond exactly to when the lights actually turned on and off each day. The microphone gain was adjusted on day 11 around hour 13, causing the last four and a half days to appear brighter in the plot. Even though the probability distributions estimated in the ELSA model are less valid after this gain change, the lighting schedule and other disturbances are still clearly visible.

The chickens were exhibiting signs of the disease most strongly from day 8 through day 11. The increased bright spots over the nighttime hours for these days correspond to increased coughing in the recordings. The amount of daytime activity in the plot also seems slightly reduced over these sick days compared to the previous days. The brightest patches in the plot during the day correspond to worker activity, such as infecting the birds, taking clinical signs, feeding, etc.

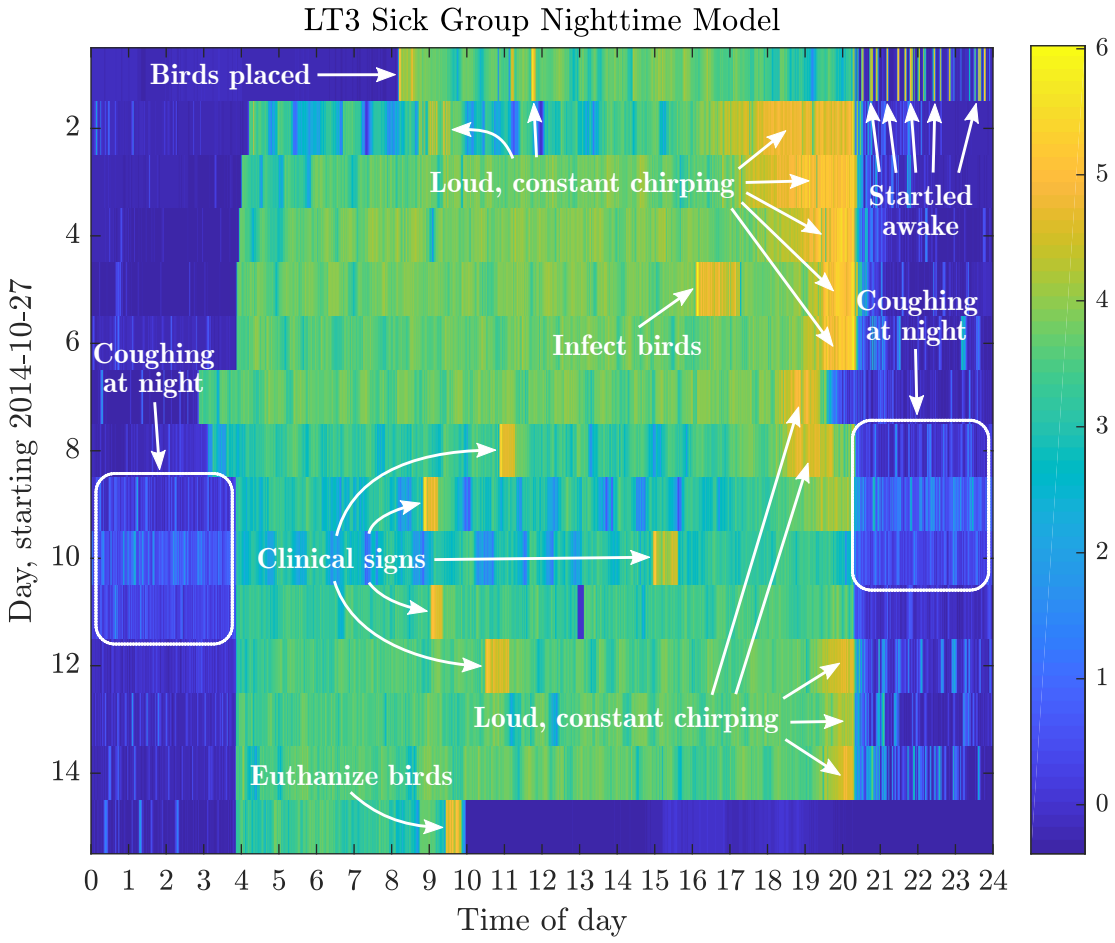

Figure 32: GAM for LT3 sick group data showing disease response. The chickens in this room were not given any vaccine, but were still infected with the LT disease. The dictionary was trained over the data prior to infection, while the ELSA model was trained over the nighttime hours from the same period.

As in Figure 31, the chickens are most sick over days 8–11, and their coughs at night cause the brighter patches during the nighttime for these days. The daytime activity also appears to be reduced during these days, matching *The Merck Veterinary Manual*'s listing of inactivity as one of the symptoms of LT [5]. Bright spots during the night on days before and after the sick period are mostly caused by general chicken activity (e.g. chirping and movement). In particular, the birds seemed to startle awake often during their first night in the room, perhaps due to their circadian rhythm adjusting to the new setting. The bright regions prior to the lights turning off on many of the days (especially the early ones) are due to constant, loud chirping activity for which we are unsure of the cause (possibly circadian rhythms). The other bright patches during the day are due to worker activities.

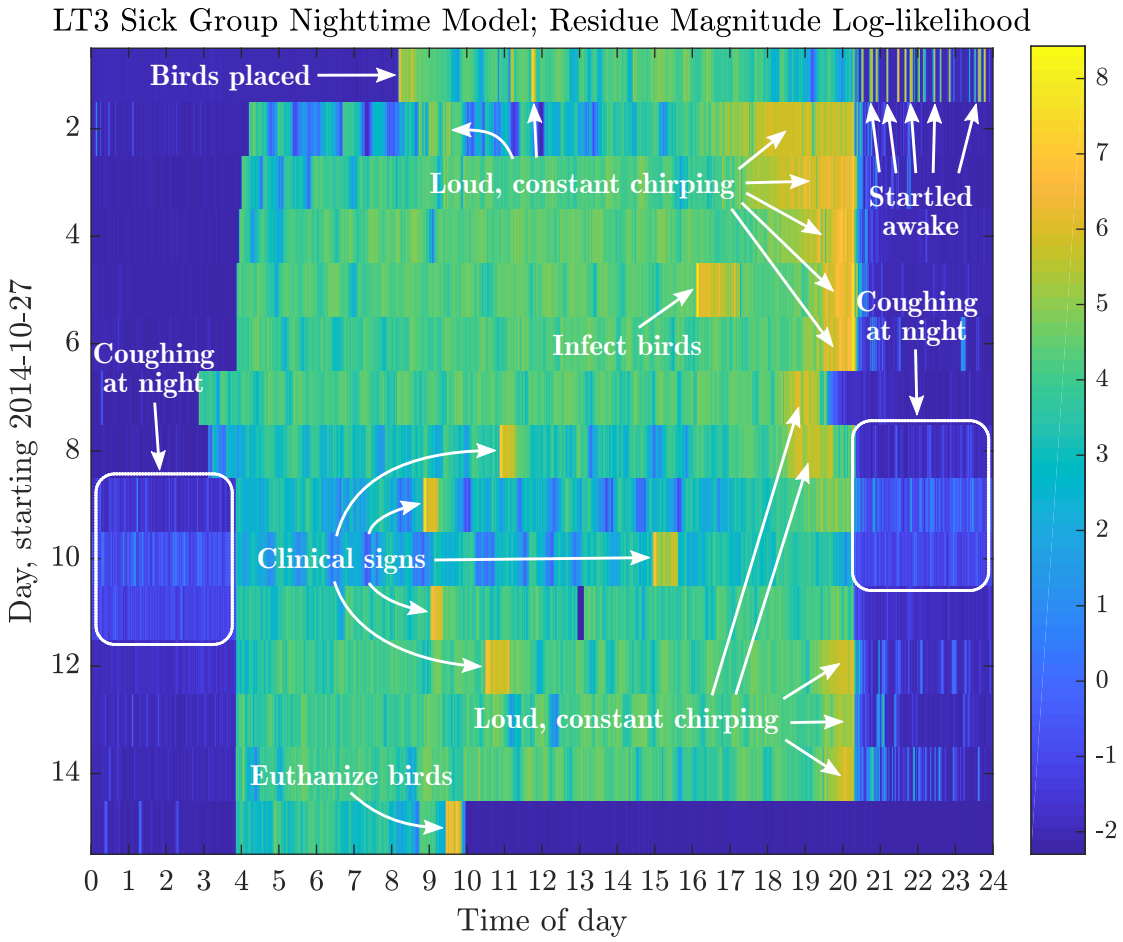

Figure 33: Residue likelihood GAM for LT3 sick group data. This plot used the same dictionary and model as Figure 32, but shows a heatmap of the log-likelihoods for the residual error instead of the average across all the random variables.

Compared to Figure 32, focusing on just the residual error likelihood from the ELSA model helps make the nighttime coughing on days 8–11 stand out a little more from the other nights. Since the dictionary was trained on a period of time when coughs were not present, it may not be able to represent them quite as well as the other nighttime activity that causes bright spots in the previous plot.

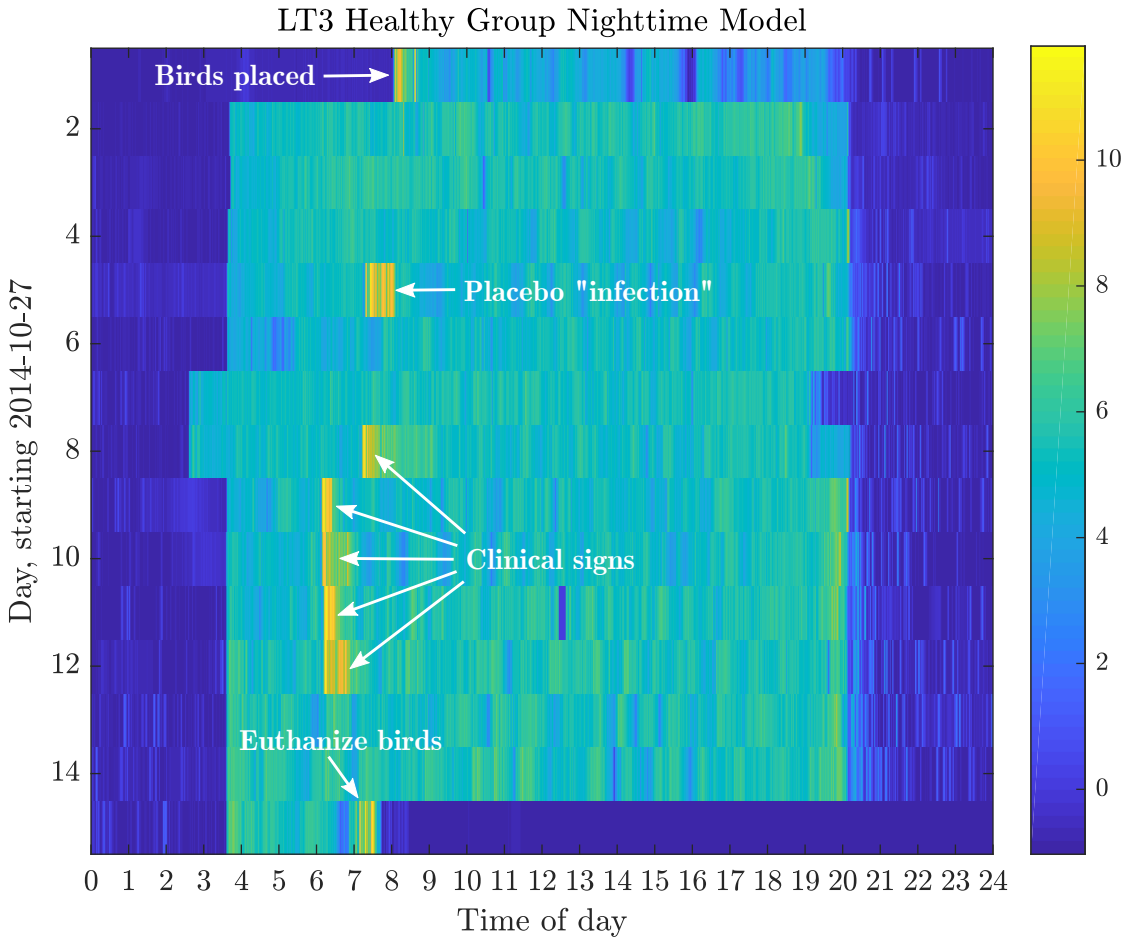

Figure 34: GAM for LT3 healthy group data. The chickens in this room were not infected with the disease. The dictionary and ELSA model were respectively trained on data prior to infection (of the other rooms) and on the corresponding nighttime data, as with the other plots.

Compared to the GAMs in Figures 31–33, note that this plot lacks the brighter bands at night and lessened daytime activity during days 8–11 when the chickens were sick in the other rooms. Bright patches are still present from when workers checked for clinical signs and performed other activities in the room.

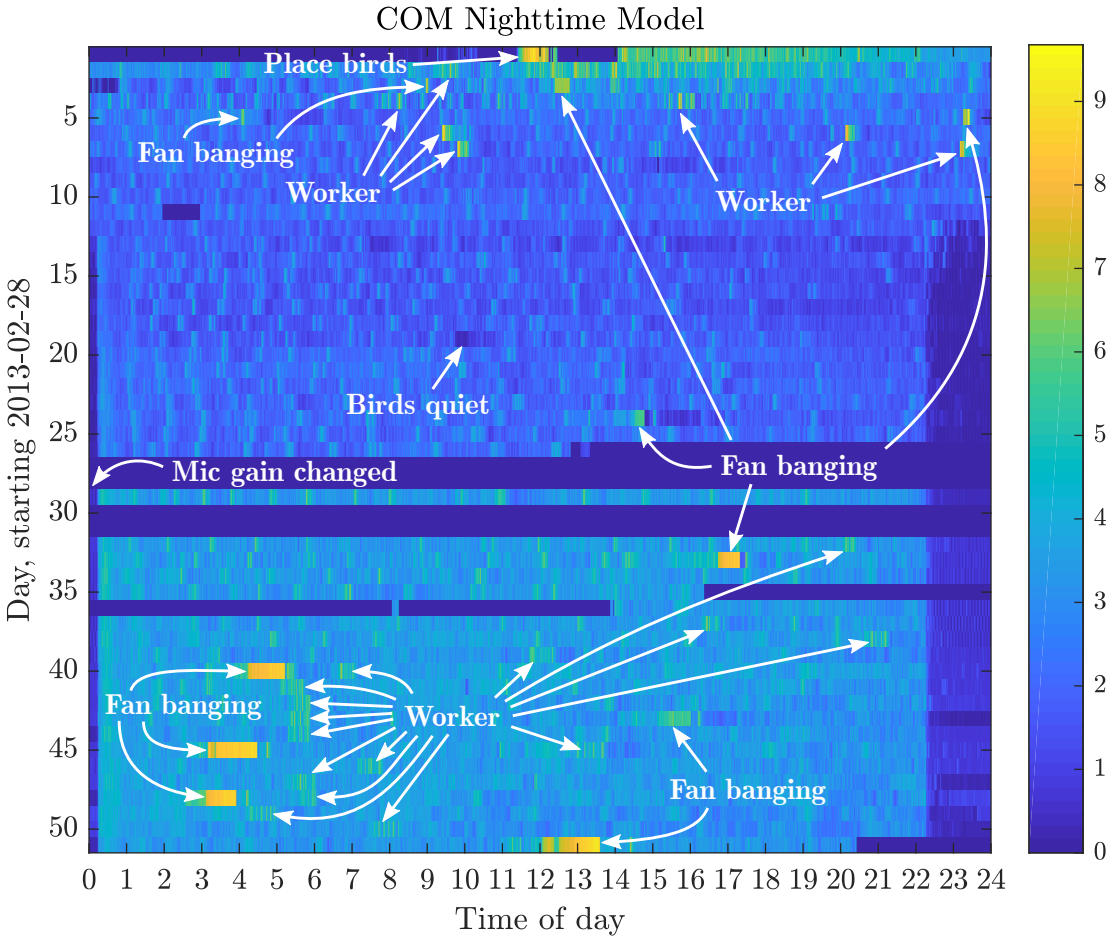

Figure 35: GAM for the COM (commercial) data. The dictionary was trained on the data from days 13–40, and the ELSA model was trained on the nighttime hours for those same days. Data is missing for a few days near the middle of the dataset.

The lights were left on constantly until the end of day 12, when they started being turned off for a couple hours each night. This is a standard practice that allows the newly hatched broiler chicks to be able to find food and water readily during a critical phase of their growth. The bottom half of the plot after the missing data is generally brighter than the top half, likely caused by a small change in the microphone gain.

The brightest segments of the plot (mostly in the bottom half) correspond to a loud mechanical clanging noise, as if a fan blade was hitting something once per rotation. This noise persists for an hour or more in several cases. Many of the other brighter spots correspond to chicken activity caused by a worker walking through the house, occasionally accompanied by comments of "Go Dawgs!" spoken into the Georgia Tech owned microphones.

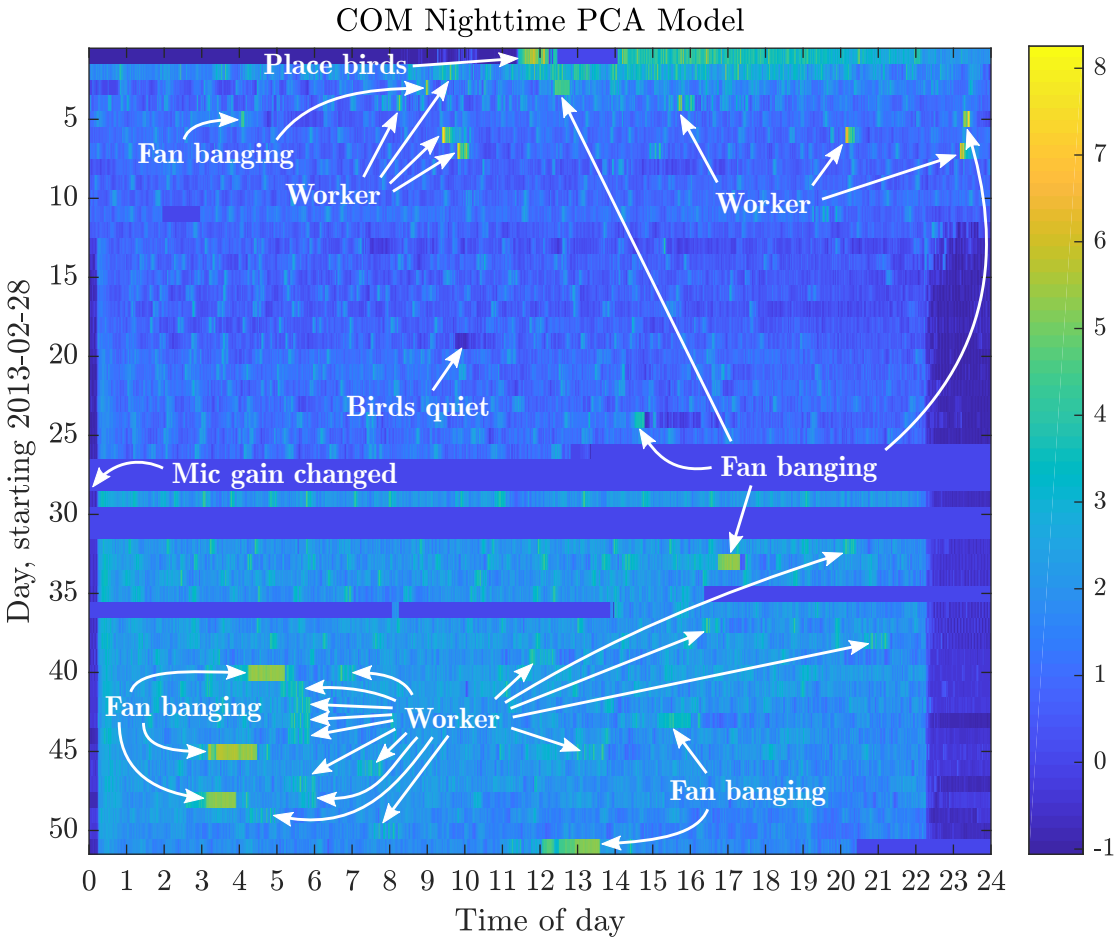

Figure 36: GAM for the COM data using the modified PCA-based method. The same regions of data were used for training as in Figure 35, but PCA and the modified PCA-based model described in Section 6.9.1 were used in place of K-SVD and the normal ELSA model.

In comparison to Figure 35, the same anomalous events are generally visible. However, there is slightly less contrast between the anomalous events and the more typical behavior around them for the PCA-based model. This matches our conclusions from Section 6.9, where the PCA-based method had similar performance to ELSA but fell a little behind in distinguishing anomalous events in daytime data.

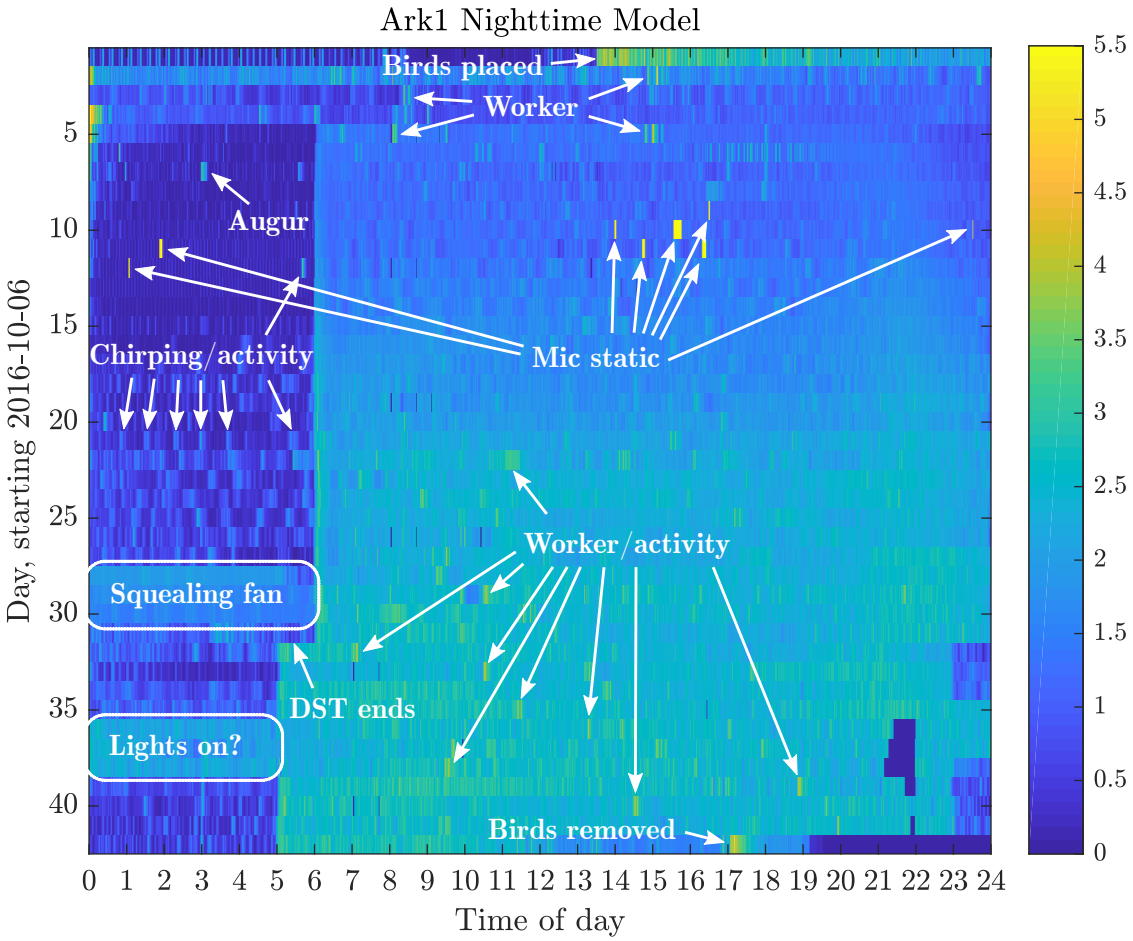

Figure 37: GAM for the Ark1 data. The dictionary was trained using data sampled from the full 42 days the chickens were present and the ELSA model was trained on data sampled from the nighttime hours.

The brightest spots on days 9–12 represent microphone malfunctions for our recording system that turned the data into loud static. As with other datasets, bright spots during the day typically correspond to chicken activity, often caused by workers passing through the house. An abnormal squealing fan noise caused the brighter band through the nighttime hours on days 28–30. The apparent one-hour shift in the lighting schedule between days 31 and 32 was due to our system updating its time for the end of daylight saving time, whereas the house controller remained on a regular schedule. We believe the lights in the house did not turn off over the three nights between days 35 and 38. The chicken activity over these nights sounds the same as typical daytime activity.

because we generally try to train dictionaries that can represent the full gamut of sounds present. If the dictionaries were smaller or trained on narrower sets of data, the atom usage and residue likelihoods might become more important.

The observation that the coefficient likelihoods tend to dominate also suggests that in some cases, it might be helpful to introduce weighting factors to adjust the relative weights between the log-likelihoods of the coefficient values, of the atoms used, and of the residue magnitudes. For some applications, it may also be appropriate to only consider one or a subset of those log-likelihoods, perhaps using them as features passed into additional learning algorithms. Many of these questions would be interesting to explore in more depth in future work.

6.12 Data explorer tool

While developing and evaluating the ELSA method, it was often difficult to match up points in our plots with the corresponding sounds in the audio. To aid in matching up the actual sound with plots of results, we developed a data explorer GUI in Matlab. The GUI presents the user with a customizable results plot that is aligned with a spectrogram plot of a given recording. The recording can be played with an animated playhead moving across both plots, and the playhead can be moved to specific times by clicking on the plots. The plots can be zoomed in and panned side-to-side along the time axis to give a finer-grained look at particular sounds. A screenshot of the GUI is shown in Figure 38.

The customizable plot in the GUI can be populated by any function that matches a defined interface. We have written several functions that allow various different aspects of the results to be plotted, such as the overall average log-likelihoods, the atom usage probabilities, the spectral features from the audio, the sparse coefficient matrix, etc. These functions typically prompt the user to select the file representing a saved ELSA model that it uses in generating the results. Some allow the user to specify multiple saved ELSA models so that their results can be plotted together for comparison.

The GUI also facilitates easy navigation of many different recordings. The table of files at the top is populated with all the recordings found in a user-specified folder. When the user

Figure 38: The Data Explorer GUI. The top pane shows a list of files in the current directory, allowing the user to switch between them with a single click. The middle pane displays a customizable plot generated by a plotting function chosen by the user. In the screenshot, a plot comparing the negative average log-likelihood values of three different ELSA models is shown. The bottom pane displays a spectrogram of the currently selected audio file. The vertical red line in both plots marks the current playback location of the audio, which can be moved with a mouse click. The view shown in the screenshot is zoomed in to the region from about 30 seconds to 55 seconds into the audio recording.

clicks on a different recording in the table, that file is automatically loaded and processed to populate the graphs and queue it up for playback. The user can easily switch from folder to folder, repopulating the table with their contents. The GUI also contains an option to normalize the volume of the files being played to make it easier to listen to recordings made at different volume levels.

6.13 Variant methods

We experimented with a few other methods prior to the development of the ELSA algorithm. While we found ELSA to be the most useful for our datasets, there are tradeoffs to these other methods that may make them more appropriate in some situations.

6.13.1 Reconstruction error

One of the first ideas we tried was to measure deviations from normal behavior by examining the magnitude of the reconstruction error between test data and its sparse approximation using a dictionary trained over normal data. Ideally, the dictionary would be able to accurately represent the types of sounds present under normal circumstances, but would not be able to represent abnormal sounds that it had not been trained on very well. This would cause the reconstruction error to be higher on data that did not fit the characterized normal behavior.

This method appeared to work well under certain limited circumstances, such as flagging coughing sounds during the night. However, it never seemed to give good results when run over an entire dataset encompassing all the complexity present in the acoustic environment of a chicken house, even when we tried counting peaks or looking at statistics of the reconstruction error over time. This is likely due to a couple factors. First, some anomalies may take the form of changes in the frequency or distribution of the sounds present rather than the introduction of new sounds that are alien to the environment. For instance, a worker passing through may cause the chickens to chirp much more frequently. Second, the magnitude of the reconstruction error will tend to be larger for louder sounds even when those sounds are normal. Since the sparse representation is neither perfectly capable of modeling normal sounds nor perfectly incapable of modeling anomalous sounds, the effects

of the loudness of the data could outweigh the effects of whether the sound was abnormal or not. When this method was applied across all the variance present in our datasets, the results seemed to correlate most strongly with the volume of the sound present rather than how anomalous it was.

To combat the effects of volume, we tried to remove it by normalizing all feature vectors to have a Euclidean magnitude of one before performing sparse coding and calculating the reconstruction error. However, this made it difficult to distinguish between random background noise and things that were important, indicating that the loudness is an important trait in recognizing anomalies.

Using the magnitude of the reconstruction error as a measure of deviation is likely to work better for detecting abnormal structures in data that can otherwise be represented very closely using sparse coding. Detecting coughs at nighttime in our data better matches these assumptions because the different equipment noises that dominate the nighttime hours tend to be very consistent, making them easier to model well with a few dictionary atoms.

We also experimented with a few different schemes for periodically retraining dictionaries to keep them relevant to the local trends in the data. However, this greatly increased the computational burden of the algorithm, and it was difficult to make comparisons between results based on different dictionaries. Instead, we have come to prefer training a single dictionary broadly so it can represent all the normal data, and then handling the local variations using different probabilistic models via the ELSA method.

6.13.2 Comparing coefficient distributions

After concluding that the reconstruction error alone did not provide sufficient information to perform the tasks we were interested in, we looked at ways to leverage the coefficient values from the sparse representation. We reasoned that if the same types of sounds are present in the observed data as are found in the normal mode of operation, then the distributions of the non-zero coefficient values for the dictionary atoms should also be similar.

Our initial approach was to estimate probability distributions for the non-zero coefficient values of both the training data and the test data, and then make a comparison between

these estimated distributions. The primary drawback of this method is that it requires a sufficiently large window of test data to get good estimates of the coefficient distributions, whereas the ELSA method can calculate the likelihood of new feature vectors immediately on a sample-by-sample basis. Setting an appropriate window size to obtain good distribution estimates is also complicated by the fact that some dictionary atoms may rarely get used under certain conditions, while other atoms will be used frequently.

For our tests, we set a constant width for the window sliding over the test data and estimated the distributions in the same way as we do for the ELSA method, as detailed in Section 6.6.2. The redistribution of the probability of one additional sample across all the bins of the distributions helped mitigate issues that may have arisen from windows with few or no samples that used a given dictionary atom. We then calculated the Bhattacharyya coefficient between the distribution for the observed data and the distribution for the training data for each atom. The coefficient is calculated as

$$BC(p, q) = \sum_{x} \sqrt{p(x)q(x)}, \tag{16}$$

where $p(x)$ and $q(x)$ are the two probability distributions under consideration [7]. The coefficient takes on a value of 1 when the two distributions are identical, and a value of 0 when there is no overlap between the two distributions. For convenience in visualizing the results, we often subtracted the Bhattacharyya coefficient from 1 to turn it into a distance measure (equivalent to the squared Hellinger distance) instead of a similarity measure.

Using this distance measure gave results similar to those previously presented for the ELSA method. We switched to the ELSA method because of the simplicity of computing the likelihoods of each sample without having to estimate distributions over a sliding window. However, this simplicity comes with a drawback.

Since the ELSA method treats each sample independently, its measure of deviation can be minimized by feeding it constant data that always uses the most likely dictionary atoms and coefficient values—even though there may have been a variety of other sounds present in the training data. To minimize the deviation measure when comparing coefficient distributions estimated over a window, the data in the test window must use a mix of

coefficient values that matches the mix seen in the training data. Thus, it provides a truer comparison than the ELSA method, but at the cost of additional complexity and more latency in obtaining results. With our data, the results were similar enough that the ELSA method was preferable.

CHAPTER 7

CONCLUSION

The research presented in this thesis makes contributions to the fields of signal processing and bioacoustics.

7.1 Signal processing contributions

The OLAF and ELSA methods represent novel and practical ways to apply sparse representations in useful ways. In particular, they reduce the need for costly efforts to label individual events within a dataset, allowing the user to leverage general knowledge of time periods when certain conditions may have been present. OLAF can help the user tease out the differences between known conditions, while ELSA can be used to measure deviations from a characterized condition.

These two methods have been tested and found to be useful in real acoustic environments that are complex, uncontrolled, and highly noisy. They are flexible in how they are trained and can make use of different types of labels or knowledge about the data. They can also be leveraged with unlabeled data to help gain an understanding of the modes present in the data and to help label it more rapidly. They both have great potential to be adapted to new situations, and lend themselves well to being used as intermediate processing steps in larger systems.

7.2 Bioacoustics contributions

The application of our methods to poultry monitoring represents a significant contribution to the field of bioacoustics. While other work has been done towards detecting different sounds from chickens or other livestock, none that we are aware of matches the scope of our work here. Most of the previous work has relied on manual extraction (and sometimes preprocessing) of the sound clips, and has involved relatively little data. Our work has dealt with continuously recorded audio over weeks and months with relatively little human input

to direct the algorithms.

Despite these differences, our algorithms have seen success in highlighting sickness, heat stress, disturbances, and abnormal equipment noise. The plots of our results also provide a useful way to visualize the trends in the behavior of the chickens over the course of a growout cycle. These methods have the potential to help characterize differences between flocks and could be useful in addressing challenges faced in the poultry industry, such as developing a better understanding of animal well-being.

7.3 Future work

There are many ways in which this work could be extended. The algorithms could be tried in various other environments where they might be applicable, such as monitoring hospital or assisted living settings. There are countless different feature types that could be used as inputs (including possible applications to non-audio data), and many parameter configurations that could be explored. Ways to make the ELSA method more robust to changes in microphone gain, such as normalizing the input features in some way, could be explored. There may also be potential to use the frozen dictionary training algorithms to help adapt parts of a dictionary to different environments while retaining information about sounds or structures that should be common between the environments. The OLAF and ELSA methods might be useful as components of larger systems or as inputs to other machine learning methods. For instance, ELSA models could potentially be treated as a type of mean and the negative log-likelihood of the data could be treated as a type of distance in a k-means-inspired algorithm that would attempt to automatically cluster or segment different conditions within a given environment.

Future work could also include efforts to develop commercial products based on these algorithms. There has been significant interest in our results from many different companies in the poultry industry. Effort would still be needed to harden the hardware to the environment and to develop interfaces that would allow farmers to interact with the algorithms.

REFERENCES

[1] Adler, A., Elad, M., Hel-Or, Y., and Rivlin, E., "Sparse coding with anomaly detection," in *Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on*, pp. 1–6, Sept. 2013.

[2] Aharon, M., Elad, M., and Bruckstein, A., "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, pp. 4311–4322, Nov. 2006.

[3] Aharon, M., Elad, M., and Bruckstein, A. M., "K-SVD and its non-negative variant for dictionary design," in *Optics & Photonics 2005*, vol. 5914, pp. 591411–591411–13, International Society for Optics and Photonics, 2005.

[4] Aide, T. M., Corrada-Bravo, C., Campos-Cerqueira, M., Milan, C., Vega, G., and Alvarez, R., "Real-time bioacoustics monitoring and automated species identification," *PeerJ*, vol. 1, p. e103, July 2013.

[5] Aiello, S. E., ed., *The Merck veterinary manual*, ch. Poultry, pp. 2379–2541. Merck, 2015. Accessed online 4/2/2015.

[6] Barchiesi, D., Giannoulis, D., Stowell, D., and Plumbley, M. D., "Acoustic scene classification: Classifying environments from the sounds they produce," *IEEE Signal Processing Magazine*, vol. 32, pp. 16–34, May 2015.

[7] Bhattacharyya, A., "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[8] Bisot, V., Serizel, R., Essid, S., and Richard, G., "Acoustic scene classification with matrix factorization for unsupervised feature learning," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6445–6449, Mar. 2016.

[9] Bogert, B. P., "The quefrency alanysis of time series for echoes ; cepstrum, pseudo-autocovariance, cross-cepstrum and saphe cracking," *Time Series Analysis*, pp. 209–243, 1963.

[10] Boracchi, G., Carrera, D., and Wohlberg, B., "Novelty detection in images by sparse representations," in *Intelligent Embedded Systems (IES), 2014 IEEE Symposium on*, pp. 47–54, Dec. 2014.

[11] Boser, B. E., Guyon, I. M., and Vapnik, V. N., "A training algorithm for optimal margin classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, (New York, NY, USA), pp. 144–152, ACM, 1992.

[12] Bradley, P. S. and Fayyad, U. M., "Refining initial points for k-means clustering," in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998)*, vol. 66, 1998.

[13] Bridle, J. S. and Brown, M. D., "An experimental automatic word-recognition system," Tech. Rep. 1003, Joint Speech Research Unity, Ruislip, England, 1974.

[14] Brown, J. C. and Smaragdis, P., "Hidden markov and gaussian mixture models for automatic call classification," *The Journal of the Acoustical Society of America*, vol. 125, no. 6, pp. EL221–EL224, 2009.

[15] Bruce, A. G., Sardy, S., and Tseng, P., "Block coordinate relaxation methods for nonparamatric signal denoising," in *Proc. SPIE*, vol. 3391, pp. 75–86, 1998.

[16] Bruckstein, A. M., Donoho, D. L., and Elad, M., "From sparse solutions of systems of equations to sparse modeling of signals and images," *SIAM Review*, vol. 51, no. 1, pp. 34–81, 2009.

[17] Canterbury, J., Struwe, F., Blankenship, E., Taira, H., and Beck, M., "Vocalizations as an indicator of distress in laying hens," in *Abstracts: 2008 Poultry Science Association*, vol. 87, p. 60, 2008.

[18] Carroll, B. T., Whitaker, B. M., Dayley, W., and Anderson, D. V., "Outlier learning via augmented frozen dictionaries," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, pp. 1207–1215, June 2017.

[19] Carroll, B. T., Britton, D. F., Daley, W., Garcia, M., Harbert, S., and Anderson, D. V., "Detection of laryngotracheitis in layer chickens via automatic nighttime cough sound detection," in *Poultry Science Association 104th Annual Meeting Abstracts*, vol. 94, 2015.

[20] Carroll, B. T., Britton, D. F., Daley, W. D., Jackwood, M. W., Harbert, S., and Anderson, D. V., "Rale detection using a microphone and audio signal processing," in *Poultry Science Association 105th Annual Meeting Abstracts*, vol. 95, 2016.

[21] Carroll, B., Anderson, D., Daley, W., Harbert, S., Britton, D., and Jackwood, M., "Detecting symptoms of diseases in poultry through audio signal processing," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pp. 1132–1135, Dec. 2014.

[22] Chandola, V., Banerjee, A., and Kumar, V., "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, July 2009.

[23] Chang, C.-C. and Lin, C.-J., "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[24] Charles, A., Olshausen, B., and Rozell, C., "Learning sparse codes for hyperspectral imagery," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 5, pp. 963–978, Sept. 2011.

[25] Chedad, A., Moshou, D., Aerts, J. M., Hirtum, A. V., Ramon, H., and Berckmans, D., "Recognition system for pig cough based on probabilistic neural networks," *Journal of Agricultural Engineering Research*, vol. 79, no. 4, pp. 449–457, 2001.

[26] Chen, S. S., Donoho, D. L., and Saunders, M. A., "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[27] Chen, Z. and Maher, R. C., "Semi-automatic classification of bird vocalizations using spectral peak tracks.," *J Acoust Soc Am*, vol. 120, pp. 2974–2984, Nov. 2006.

[28] Cheng, J., Sun, Y., and Ji, L., "A call-independent and automatic acoustic system for the individual recognition of animals: A novel model using four passerines," *Pattern Recognition*, vol. 43, pp. 3846–3852, Nov. 2010.

[29] Choi, S., Cichocki, A., Park, H.-M., and Lee, S.-Y., "Blind source separation and independent component analysis: A review," *Neural Information Processing - Letters and Reviews*, vol. 6, no. 1, 2005.

[30] Chu, S., Narayanan, S., and Kuo, C.-C. J., "Environmental sound recognition with time-frequency audio features," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 17, pp. 1142–1158, Aug. 2009.

[31] Chung, Y., Lee, J., Oh, S., Park, D., Chang, H., and Kim, S., "Automatic detection of cow's oestrus in audio surveillance system," *Asian-Australasian journal of animal sciences*, vol. 26, no. 7, p. 1030, 2013.

[32] Chung, Y., Oh, S., Lee, J., Park, D., Chang, H.-H., and Kim, S., "Automatic detection and recognition of pig wasting diseases using sound data in audio surveillance systems," *Sensors*, vol. 13, no. 10, pp. 12929–12942, 2013.

[33] Clemins, P. J., *Automatic classification of animal vocalizations*. PhD thesis, Marquette University, Milwaukee, Wisconsin, May 2005.

[34] Clemins, P. J., Johnson, M. T., Leong, K. M., and Savage, A., "Automatic classification and speaker identification of african elephant (loxodonta africana) vocalizations," *The Journal of the Acoustical Society of America*, vol. 117, no. 2, pp. 956–963, 2005.

[35] Clemins, P. J., Trawicki, M. B., Adi, K., Tao, J., and Johnson, M. T., "Generalized perceptual features for vocalization analysis across multiple species," in *Acoustics, Speech and Signal Processing, 2006. ICASSP 2006 Proceedings. 2006 IEEE International Conference on*, vol. 1, May 2006.

[36] Colón, G. J., "Avian musing feature space analysis," Master's thesis, Georgia Institute of Technology, 2012.

[37] Cong, Y., Yuan, J., and Liu, J., "Abnormal event detection in crowded scenes using sparse representation," *Pattern Recognition*, vol. 46, no. 7, pp. 1851 – 1864, 2013.

[38] Cortes, C. and Vapnik, V., "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[39] Curtin, R., Daley, W., and Anderson, D., "Classifying broiler chicken condition using audio data," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pp. 1141–1144, Dec. 2014.

[40] Davis, S. and Mermelstein, P., "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 28, pp. 357–366, Aug. 1980.

[41] Donoho, D. L., "For most large underdetermined systems of linear equations the minimal $\ell 1$-norm solution is also the sparsest solution," *Communications on Pure and Applied Mathematics*, vol. 59, no. 6, pp. 797–829, 2006.

[42] Donoho, D., Elad, M., and Temlyakov, V., "Stable recovery of sparse overcomplete representations in the presence of noise," *Information Theory, IEEE Transactions on*, vol. 52, pp. 6–18, Jan. 2006.

[43] Donoho, D., Tsaig, Y., Drori, I., and Starck, J.-L., "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *Information Theory, IEEE Transactions on*, vol. 58, pp. 1094–1121, Feb. 2012.

[44] Engan, K., Aase, S. O., and Husøy, J. H., "Multi-frame compression: theory and design," *Signal Processing*, vol. 80, no. 10, pp. 2121 – 2140, 2000.

[45] Eronen, A. J., Peltonen, V. T., Tuomi, J. T., Klapuri, A. P., Fagerlund, S., Sorsa, T., Lorho, G., and Huopaniemi, J., "Audio-based context recognition," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, pp. 321–329, Jan. 2006.

[46] Fagerlund, S., "Bird species recognition using support vector machines," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, pp. 1–8, 2007.

[47] Gorodnitsky, I. and Rao, B., "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *Signal Processing, IEEE Transactions on*, vol. 45, pp. 600–616, Mar. 1997.

[48] Harma, A., "Automatic identification of bird species based on sinusoidal modeling of syllables," in *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, vol. 5, pp. V–545–8 vol.5, Apr. 2003.

[49] Hotelling, H., "Analysis of a complex of statistical variables into principal components.," *Journal of educational psychology*, vol. 24, no. 6, p. 417, 1933.

[50] Hoyer, P., "Non-negative sparse coding," in *Neural Networks for Signal Processing, 2002. Proceedings of the 2002 12th IEEE Workshop on*, pp. 557–565, 2002.

[51] Huang, K. and Aviyente, S., "Sparse representation for signal classification," in *Advances in neural information processing systems*, pp. 609–616, 2006.

[52] Jahns, G., "Call recognition to identify cow conditions—a call-recogniser translating calls to text," *Computers and Electronics in Agriculture*, vol. 62, pp. 54–48, June 2008.

[53] Jankowski Jr, C. R., Vo, H.-D., and Lippmann, R. P., "A comparison of signal processing front ends for automatic word recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 4, pp. 286–293, 1995.

[54] Jutten, C. and Herault, J., "Blind separation of sources, part I: An adaptive algorithm based on neuromimetic architecture," *Signal Processing*, vol. 24, no. 1, pp. 1 – 10, 1991.

[55] Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A., "An efficient k-means clustering algorithm: analysis and implementation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, pp. 881–892, July 2002.

[56] Kogan, J. A. and Margoliash, D., "Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden Markov models: A comparative study," *The Journal of the Acoustical Society of America*, vol. 103, no. 4, pp. 2185–2196, 1998.

[57] Koh, K., Kim, S.-J., and Boyd, S. P., "An interior-point method for large-scale l1-regularized logistic regression.," *Journal of Machine learning research*, vol. 8, no. 8, pp. 1519–1555, 2007.

[58] Kohlsdorf, D., *Data mining in large audio collections of dolphin signals*. PhD thesis, Georgia Institute of Technology, July 2015.

[59] Kreutz-Delgado, K., Murray, J. F., Rao, B. D., Engan, K., Lee, T.-W., and Sejnowski, T. J., "Dictionary learning algorithms for sparse representation," *Neural computation*, vol. 15, no. 2, pp. 349–396, 2003.

[60] Lee, C.-H., Chou, C.-H., Han, C.-C., and Huang, R.-Z., "Automatic recognition of animal vocalizations using averaged MFCC and linear discriminant analysis," *Pattern Recognition Letters*, vol. 2, pp. 93–101, Jan. 2006.

[61] Lee, D. D. and Seung, H. S., "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[62] Lee, D. D. and Seung, H. S., "Algorithms for non-negative matrix factorization," in *Advances in Neural Information Processing Systems 13* (Leen, T., Dietterich, T., and Tresp, V., eds.), pp. 556–562, MIT Press, 2001.

[63] Lee, J., Noh, B., Jang, S., Park, D., Chung, Y., and Chang, H.-H., "Stress detection and classification of laying hens by sound analysis.," *Asian-Australasian journal of animal sciences*, 2015.

[64] Lee, J., Zuo, S., Chung, Y., Park, D., Chang, H.-H., and Kim, S., "Formant-based acoustic features for cow's estrus detection in audio surveillance system," in *Advanced Video and Signal Based Surveillance (AVSS), 2014 11th IEEE International Conference on*, pp. 236–240, IEEE, 2014.

[65] Lesage, S., Gribonval, R., Bimbot, F., and Benaroya, L., "Learning unions of orthonormal bases with thresholded singular value decomposition," in *Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05). IEEE International Conference on*, vol. 5, pp. 293–296, Mar. 2005.

[66] Lewicki, M. S. and Olshausen, B. A., "Probabilistic framework for the adaptation and comparison of image codes," *J. Opt. Soc. Am. A*, vol. 16, pp. 1587–1601, July 1999.

[67] Li, C., Han, Z., Ye, Q., and Jiao, J., "Abnormal behavior detection via sparse reconstruction analysis of trajectory," in *Image and Graphics (ICIG), 2011 Sixth International Conference on*, pp. 807–810, Aug. 2011.

[68] Lloyd, S., "Least squares quantization in PCM," *Information Theory, IEEE Transactions on*, vol. 28, pp. 129–137, Mar. 1982.

[69] MacQueen, J., "Some methods for classification and analysis of multivariate observations," *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[70] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online dictionary learning for sparse coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, (New York, NY, USA), pp. 689–696, ACM, 2009.

[71] Mairal, J., Bach, F., Ponce, J., and Sapiro, G., "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 19–60, Mar. 2010.

[72] Mallat, S. and Zhang, Z., "Matching pursuits with time-frequency dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, pp. 3397–3415, Dec. 1993.

[73] Manteuffel, G., Puppe, B., and Schon, P. C., "Vocalization of farm animals as a measure of welfare," *Applied Animal Behaviour Science*, vol. 88, pp. 163–182, Sept. 2004.

[74] Manteuffel, G. and Schön, P. C., "Measuring pig welfare by automatic monitoring of stress calls," *Bornimer Agrartech. Ber*, vol. 29, pp. 110–118, 2002.

[75] Marx, G., Horn, T., Thielebein, J., Knubel, B., and von Borell, E., "Analysis of pain-related vocalization in young pigs," *Journal of Sound and Vibration*, vol. 266, no. 3, pp. 687 – 698, 2003. First International {ISMA} Workshop on Noise and Vibration in Agricultural and Biological Engineering.

[76] Matos, S., Birring, S., Pavord, I., and Evans, D., "Detection of cough signals in continuous audio recordings using hidden Markov models," *Biomedical Engineering, IEEE Transactions on*, vol. 53, pp. 1078–1083, June 2006.

[77] Mermelstein, P., "Distance measures for speech recognition—psychological and instrumental," in *Pattern Recognition and Artificial Intelligence* (Chen, C. H., ed.), (New York), pp. 374–388, Academic Press, Inc., 1976.

[78] Mo, X., Monga, V., Bala, R., and Fan, Z., "Adaptive sparse representations for video anomaly detection," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 24, pp. 631–645, Apr. 2014.

[79] Moura, D., Silva, W., Naas, I., Tolon, Y., Lima, K., and Vale, M., "Real time computer stress monitoring of piglets using vocalization analysis," *Computers and Electronics in Agriculture*, vol. 64, pp. 11–18, Nov. 2008.

[80] Natarajan, B. K., "Sparse approximate solutions to linear systems," *SIAM Journal on Computing*, vol. 24, no. 2, pp. 227–234, 1995.

[81] National Agricultural Statistics Service, "Poultry - production and value, 2013 summary," tech. rep., United States Department of Agriculture, Apr. 2014.

[82] Olshausen, B. A. and Field, D. J., "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, no. 6583, pp. 607–609, 1996.

[83] Olshausen, B. A. and Field, D. J., "Sparse coding with an overcomplete basis set: A strategy employed by V1?," *Vision Research*, vol. 37, no. 23, pp. 3311 – 3325, 1997.

[84] O'Shaughnessy, D., *Speech communication: human and machine*. Addison-Wesley, 1987.

[85] Otu-Nyarko, E., Darre, M., Scheifele, P., Miller, D., and Johnson, M., "Classification of stressful vocalizations of captive laying chickens using the hidden Markov model (HMM)," in *Abstracts: 2008 Poultry Science Association*, vol. 87, pp. 59–60, 2008.

[86] Otu-Nyarko, E., *The effect of stress on the vocalizations of captive poultry populations*. PhD thesis, University of Connecticut, 2010.

[87] Parrish, N. V., Anderson, D. V., Daley, W. D. R., Britton, D. F., Harbert, S., Webster, A. B., and Ritz, C. W., "Analysis of poultry vocalizations for age estimation using cepstral coefficients and C4.5 classification," in *Proceedings of the American Society of Agricultural and Biological Engineers*, American Society of Agricultural and Biological Engineers, 2013.

[88] Pati, Y., Rezaiifar, R., and Krishnaprasad, P., "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*, pp. 40–44 vol.1, Nov. 1993.

[89] Pearson, K., "On lines and planes of closest fit to systems of points in space," *Philosophical Magazine Series 6*, vol. 2, no. 11, pp. 559–572, 1901.

[90] Pimentel, M. A., Clifton, D. A., Clifton, L., and Tarassenko, L., "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215 – 249, 2014.

[91] Platt, J., "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft Research, Apr. 1998.

[92] Pols, L. C. W., *Spectral analysis and identification of Dutch vowels in monosyllabic words*. PhD thesis, University of Amsterdam, 1977.

[93] Quinlan, J. R., *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[94] Quinlan, J., "Induction of decision trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986.

[95] Ren, Y., Johnson, M. T., Clemins, P. J., Darre, M., Glaeser, S. S., Osiejuk, T. S., and Out-Nyarko, E., "A framework for bioacoustic vocalization analysis using hidden Markov models," *Algorithms*, vol. 2, no. 4, p. 1410, 2009.

[96] Rubinstein, R., Zibulevsky, M., and Elad, M., "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS Technion*, vol. 40, no. 8, pp. 1–15, 2008.

[97] Sandhu, S. and Ghitza, O., "A comparative study of mel cepstra and EIH for phone classification under adverse conditions," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1, pp. 409–412, IEEE, 1995.

[98] Smaragdis, P., "From learning music to learning to separate," in *Forum Acusticum*, Citeseer, 2005.

[99] Smaragdis, P., "Probabilistic decompositions of spectra for sound separation," in *Blind Speech Separation* (Makino, S., Sawada, H., and Lee, T.-W., eds.), Signals and Communication Technology, pp. 365–386, Springer Netherlands, 2007.

[100] Smith, L. N. and Elad, M., "Improving dictionary learning: Multiple dictionary updates and coefficient reuse," *Signal Processing Letters, IEEE*, vol. 20, no. 1, pp. 79–82, 2013.

[101] Somervuo, P. and Harma, A., "Bird song recognition based on syllable pair histograms," in *Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04). IEEE International Conference on*, vol. 5, pp. 825–8, May 2004.

[102] Stevens, S. S. and Volkmann, J., "The relation of pitch to frequency: A revised scale," *The American Journal of Psychology*, vol. 53, no. 3, pp. 329–353, 1940.

[103] Stevens, S. S., Volkmann, J., and Newman, E. B., "A scale for the measurement of the psychological magnitude pitch," *The Journal of the Acoustical Society of America*, vol. 8, no. 3, pp. 185–190, 1937.

[104] Tropp, J., "Greed is good: algorithmic results for sparse approximation," *Information Theory, IEEE Transactions on*, vol. 50, pp. 2231–2242, Oct. 2004.

[105] University of Kentucky, "Poultry production manual." `http://www2.ca.uky.edu/poultryprofitability/production_manual.html`, 2015. Chapter 16. Accessed online 11/7/2015.

[106] Valente, D., Wang, H., Andrews, P., Mitra, P. P., Saar, S., Tchernichovski, O., Golani, I., and Benjamini, Y., "Characterizing animal behavior through audio and video signal processing," *MultiMedia, IEEE*, vol. 14, pp. 32–41, Oct. 2007.

[107] Vandermeulen, J., Bahr, C., Tullo, E., Fontana, I., Ott, S., Kashiha, M., Guarino, M., Moons, C. P. H., Tuyttens, F. A. M., Niewold, T. A., and Berckmans, D., "Discerning pig screams in production environments," *PLoS ONE*, vol. 10, p. e0123111, Apr. 2015.

[108] Whitaker, B., Carroll, B., Daley, W., and Anderson, D., "Sparse decomposition of audio spectrograms for automated disease detection in chickens," in *Signal and Information Processing (GlobalSIP), 2014 IEEE Global Conference on*, pp. 1122–1126, Dec. 2014.

[109] Wright, J., Yang, A., Ganesh, A., Sastry, S., and Ma, Y., "Robust face recognition via sparse representation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, pp. 210–227, Feb. 2009.

[110] Zhao, B., Fei-Fei, L., and Xing, E., "Online detection of unusual events in videos via dynamic sparse coding," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 3313–3320, June 2011.

[111] Zhou, J., Semenovich, D., Sowmya, A., and Wang, J., "Sparse dictionary reconstruction for textile defect detection," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, vol. 1, pp. 21–26, Dec. 2012.

[112] Zimek, A., Schubert, E., and Kriegel, H.-P., "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining*, vol. 5, no. 5, pp. 363–387, 2012.